



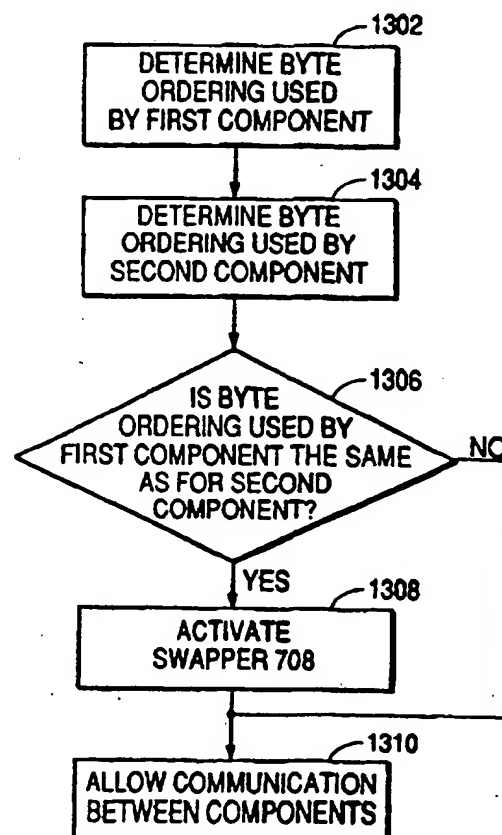
## INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification <sup>5</sup> : <b>G06F</b>	<b>A2</b>	(11) International Publication Number: <b>WO 94/15269</b> (43) International Publication Date: <b>7 July 1994 (07.07.94)</b>
--	-----------	---

(21) International Application Number: **PCT/US93/12416**(22) International Filing Date: **17 December 1993 (17.12.93)**(30) Priority Data:  
**07/994,405**      **21 December 1992 (21.12.92)**      **US**(71) Applicant: **OLIVETTI ADVANCED TECHNOLOGY CENTER, INC. [US/US]; 20300 Stevens Creek Boulevard, Cupertino, CA 95014 (US).**(72) Inventor: **TAM, Stephen, Yiu-Ming; 2858 Dominici Drive, Fremont, CA 94536 (US).**(74) Agents: **McKIE, Edward, F., Jr. et al.; Banner, Birch, McKie & Beckett, 1001 G Street, N.W., 11th floor, Washington, DC 20001-4597 (US).**(81) Designated States: **JP, European patent (AT, BE, CH, DE, DK, ES, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).****Published***Without international search report and to be republished upon receipt of that report.*(54) Title: **APPARATUS, SYSTEM AND METHOD FOR FACILITATING COMMUNICATION BETWEEN COMPONENTS HAVING DIFFERENT BYTE ORDERINGS**

## (57) Abstract

A system and method for allowing a component having a first byte ordering to effectively transfer information to another component having a second byte ordering. The present invention envisions embodiments where facilitation of the information transmission is set depending upon whether the two components have the same byte ordering or whether they have different byte orderings.



**FOR THE PURPOSES OF INFORMATION ONLY**

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AT	Austria	GB	United Kingdom	MR	Mauritania
AU	Australia	GE	Georgia	MW	Malawi
BB	Barbados	GN	Guinea	NE	Niger
BE	Belgium	GR	Greece	NL	Netherlands
BF	Burkina Faso	HU	Hungary	NO	Norway
BG	Bulgaria	IE	Ireland	NZ	New Zealand
BJ	Benin	IT	Italy	PL	Poland
BR	Brazil	JP	Japan	PT	Portugal
BY	Belarus	KE	Kenya	RO	Romania
CA	Canada	KG	Kyrgyzstan	RU	Russian Federation
CF	Central African Republic	KP	Democratic People's Republic of Korea	SD	Sudan
CG	Congo	KR	Republic of Korea	SE	Sweden
CH	Switzerland	KZ	Kazakhstan	SI	Slovenia
CI	Côte d'Ivoire	LI	Liechtenstein	SK	Slovakia
CM	Cameroon	LK	Sri Lanka	SN	Senegal
CN	China	LU	Luxembourg	TD	Chad
CS	Czechoslovakia	LV	Latvia	TG	Togo
CZ	Czech Republic	MC	Monaco	TJ	Tajikistan
DE	Germany	MD	Republic of Moldova	TT	Trinidad and Tobago
DK	Denmark	MG	Madagascar	UA	Ukraine
ES	Spain	ML	Mali	US	United States of America
FI	Finland	MN	Mongolia	UZ	Uzbekistan
FR	France			VN	Viet Nam
GA	Gabon				

-1-

**APPARATUS, SYSTEM AND METHOD FOR  
FACILITATING COMMUNICATION BETWEEN  
COMPONENTS HAVING DIFFERENT BYTE ORDERINGS**

5

**BACKGROUND OF THE INVENTION**

**I. Field of the Invention**

10

This invention relates to a system and method for allowing a computer-related component having a first byte ordering to effectively communicate with a computer-related component having a second byte ordering. Embodiments of the present invention also envision a system and method for determining whether two components have the same byte ordering and for making provisions allowing the two components to communicate with each other. In addition, embodiments of the present invention contemplate that these provisions allowing the two components to communicate with each other can be administered dynamically.

15

20

**II. Related Art**

25

Over the past several decades, various types of computer architectures have evolved. Many computer manufacturers have developed their own schemes and protocols, often stifling communication between devices using different schemes and protocols. Consequently, much effort has been expended in the computer industry in recent years trying to allow otherwise "incompatible" devices to communicate with each other.

30

One such protocol difference that has developed between computer manufacturers concerns the "byte

-2-

ordering" within a computer system. Byte ordering relates to the order in which bytes are ranked (typically within a "word" or portion thereof) from least significant to most significant in relation to the bit order (which concerns the order in which bits are ranked).

Components such as processor interfaces, memory devices and I/O interfaces typically need to be pre-configured to utilize a particular byte ordering. This is so these components can effectively communicate within a conventional computer system. Software components such as operating systems and application programs typically inherently utilize (and can be said to "have") the byte ordering of the hardware they were created on. Byte ordering is not an issue for the registers of a processor itself, since the bytes within a register cannot typically be selectively accessed (and thus, by default, the least significant byte is the one comprising the lowest order bits).

The majority of computer architectures utilize one of two different types of byte ordering schemes. In the first type, the least significant byte comprises the eight least-significant bits of a word. Similarly, the most significant byte comprises the eight most-significant bits. The significance of each byte is thus proportional to the significance of the bits that it encompasses.

This first type of byte ordering is known in the industry as "little endian" byte ordering. This terminology originates from a publication entitled "On Holy Wars And A Plea For Peace" by Danny Cohen (IEEE Computer, October, 1981). Little endian byte ordering has been adopted by such computer manufacturers as the Digital Equipment Corporation of Maynard, Massachusetts in their VAX line of computers and by computers using

-3-

processors manufactured by Intel Corporation of Santa Clara, California.

5 In the second type of byte ordering, the least significant byte comprises the eight most-significant bits of a word. The significance of each byte is thus inversely proportional to the significance of the bits that it comprises. This type of byte ordering is known as "big endian" byte ordering and has been adopted by  
10 International Business Machines of Armonk, N.Y. in their 370 mainframe computers and in computers using 68000 line of processors manufactured by Motorola of Schaumburg, Ill. It should be noted that for both big and little endian types of byte ordering, the  
15 significance of the bits (i.e., the bit order) is unaffected by the byte ordering used.

The type of byte ordering utilized by components within a particular computer system affects the way that information is transmitted between those  
20 components, particularly when less than a word is transmitted at any given time. The specific way this information is transmitted (usually via a bus of some type) is dictated by convention and does not follow a readily apparent pattern. For this reason, examples of information transmission in both a big endian computer  
25 system (i.e., a computer system using components having a big endian byte ordering) and little endian computer system are discussed below with regard to Figures 1-3.

30 Referring first to Figure 1, this Figure shows the effect of a single byte of information transmitted from a register 102 within a processor (not shown) across the busses of two different computer systems (each having its own processor and register 102). Specifically, the information is transmitted across a  
35 32 bit bus 104 within a little endian computer system

and across a 32 bit bus 106 within a big endian computer system. Since only one byte is being transmitted, the information is positioned in the 8 least-significant bits of register 102. The register 102 is thus presumed to be within a processor having a little endian byte ordering in the example using bus 104 and in a processor having a big endian byte ordering in the example using bus 106.

In Figure 1, the busses 104 and 106 are labeled "byte 0" - "byte 3." This represents the significance of each byte (with byte 0 being the least significant byte) transmitted across the busses. Here, the single byte of information (information (A)) from register 102 is shown (for each computer system) being transmitted at "byte 1" across each bus to a location representative of "byte 1" of some component (not shown). The busses 104 and 106 thus reflect the byte ordering that the bytes are received by the component. Consequently, the way that byte information is transmitted on the busses 104 and 106 can be used as a good illustration of the differences between the use of big endian and little endian byte ordering.

The most apparent difference between the little endian byte ordering as shown by bus 104 and the big endian byte ordering as shown by bus 106 is that for the little endian byte ordering byte 0 comprises the 8 least-significant bits while for the big endian byte ordering byte 0 comprises the 8 most-significant bits. Thus, as stated above, for little endian byte ordering the byte ordering is directly proportional to the bit ordering while for big endian byte ordering the byte and bit ordering are inversely proportional.

Therefore, in the little endian byte ordering example shown by bus 104, the byte of information (A) appears at byte 1 along the bus 104. Here, byte 1

-5-

comprises bits 8-15. In the big endian byte ordering example shown by bus 106, the byte of information (A) also appears at the byte 1 position along the bus 106. However, now byte 1 encompasses bits 16-23.

Consequently, the little endian byte ordering and big endian byte ordering appear to have been "flipped" in this example.

An example of the effect of transmitting a half-word (2 bytes) of information from register 102 across busses 104 and 106 is now explained with regard to Figure 2. Referring to Figure 2, register 102 is shown to contain a half-word of information which is to be transmitted to bytes 0 and 1 of some component (not shown) across busses 104 and 106 for each respective computer system. In the little endian computer system represented by bus 104, the first byte in the register (information (A)) appears at byte 0 and the second byte in the register (information (B)) appears at byte 1. Thus, information (A) is transmitted across the 8 least-significant bits and information (B) is transmitted across the 8 next least-significant bits (i.e., bits 8-15).

In contrast, in the big endian computer system (represented by bus 106) information (A) appears at byte 1 (shown as comprising bits 23-16) and information (B) appears at byte 0 (comprising the 8-most significant bits). Note that the information across bus 106 has been "shifted" rather than "flipped" as compared with the information across bus 104 (i.e., byte 0 in the little endian byte ordering scheme contains information (A) whereas byte 1, rather than byte 0, in the big endian byte ordering scheme contains information (A)).

Figure 3 discloses the effect of a word (in this example, 4 bytes) of information being transmitted

across busses 104 and 106. Referring now to Figure 3, it can be seen that the order of information in register 102 is the same as the order of information across bus 104 representing the little endian byte ordering. Thus, information (A) comprises the 8 least-significant bits in both register 102 and across bus 104.

For the big endian byte ordering across bus 106, a word of information is transmitted in the same bit sequence as for the little endian byte ordering across bus 104. Thus, when a word is transmitted across bus 106, information (A), for example, is transmitted across the 8 least-significant bits, the same as it was for bus 104. Note, however, that the order of what each byte ordering considers as byte 0, byte 1, etc., never changes.

An additional explanation of the byte ordering of little endian and big endian protocols can be found in "On Holy Wars And A Plea For Peace" described above. Also, while the above-mentioned examples depict a 32-bit computer architecture, the various concepts described apply to architectures of other sizes as well.

When operating within a conventional computing environment where all components (processors, memory, etc.) utilize the same byte ordering, the byte ordering is typically of no concern to the average user or programmer in day-to-day operations. This is because when byte-sized information is written to a particular address, it can then typically be accessed at that same address. This is a fundamental principle without which computers would find it almost impossible to function.

Because of incompatibility problems between the two byte ordering schemes (which will be explained below), the byte ordering of a particular computer restricts

-7-

the d cisions that a user can make when deciding which types of software to buy. More specifically, operating systems are typically written for either one byte ordering or another. For example, Microsoft NT has been designed for use with little endian computer systems while the majority of Unix operating systems have been designed for big endian computer systems. Consequently, the type of operating system which a user wants to implement drives the decision for the type of computer system that the user can purchase. Looked at from another perspective, if a user has already purchased a computer, the options for purchase of an operating system are limited.

Thus, a need has arisen for a computer system which can run an operating system (and any software executing thereon) utilizing either big endian or little endian byte ordering. In this way, users could purchase such a computer system and then use software of either byte ordering. Thus, the user would be able to implement either a big endian or little endian operating system since the computer system would be capable of operating in using both byte ordering schemes.

A key step in designing a computer system which can execute software utilizing either big endian or little endian byte ordering is to design a processor that can operate in either a big endian or little endian mode. MIPS Computer Systems of Sunnyvale, California has designed a microprocessor (the R-4000) which is bi-endian (i.e., capable of operating in either a big endian and a little endian mode). Consequently, operating systems using either big endian or little endian byte ordering can be executed using this microprocessor.

Howev r, it is not enough to mer ly create a processor which is bi-endian. This is because

conventional computer components are designed to utilize either big endian or little endian byte ordering. That is, components such as memory devices are configured to be properly addressable in conjunction with either a big endian byte ordering scheme or a little endian byte ordering scheme.

Though it may be possible to build a computer system where every component is bi-endian, it would be more cost-effective to build a computer system using a bi-endian processor and existing conventional components. To build such a system, the computer system designer would have to choose either big endian or little endian components (that is, components having either a big endian or a little endian byte ordering). The problem arises when the bi-endian processor is set to operate in a first byte ordering mode and a component that it exchanges information with has a second byte ordering mode. An example of the problems encountered in this situation can be best explained with reference to the example of Figure 4.

Referring now to Figure 4, a bi-endian processor 402 is shown operating in a big endian mode. Thus, at a bus interface 408 of the bi-endian processor 402, byte 0 encompass the 8 most-significant bits. In this particular example, a single byte of information (A) is shown being transmitted across bus 404.

Also shown attached to bus 404 is a little endian component 406. This component could be, for example, a memory device.

When the bi-endian processor 402 transmits information (A) onto bus 404 at byte 0, a signal is also sent to the receiving component 406 at byte 0. If the receiving component 406 had a big endian byte ordering, then the transaction would have the expected results of enabling the portion of the component 406

-9-

that receives information (A). However, in this example, byte 0 of the component 406 is enabled (i.e., receptive to information), but information (A) is received at byte 3 (which is not enabled).

5 More specifically, because of industry-standard designs, when a byte is transmitted onto a bus from a particular byte location (e.g., byte 0) the corresponding byte location of the receiving component is enabled. Because of this and standard bus designs,  
10 information (A) will not be received by any portion of little endian component 406 at all.

The same type of problem described above would occur if a half-word were transmitted to the little endian component 406. However, if an entire word is  
15 transmitted, then as can be envisioned with the additional aid of Figure 3, all four bytes of the little endian component 406 would be enabled and thus all of the information would be received. Specifically, information which is at the bus interface  
20 408 at byte 0 of the bi-endian processor 402 would be received by byte 3 of little endian component 406, byte 1 of the bus interface 408 would be written to byte 2 of little endian component 406, etc. If the entire word were then read back from the little endian  
25 component 406, it would be received by bus interface 402 of the bi-endian processor 402 in the same order as it had been transmitted in.

As can be appreciated by the seemingly convoluted nature of the big endian and little endian byte  
30 ordering schemes, creating a solution to allow big endian components to effectively communicate with little endian components for the multitude of possible byte transmission situations is no simple task. One solution to this problem has been disclosed by MIPS  
35 Computer Systems in their European patent application

-10-

number 47057082. In this application, the byte addresses of the target component (i.e., the component which receives information) are translated so that information can be received. More specifically and using the example shown in Figure 4, when information (A) from byte 0 is transmitted across data bus 404, the receiving byte address for the little endian component would be translated from byte 0 over to byte 3. Consequently, byte 3 of the little endian component 406 is enabled (rather than byte 0) and thus byte 3 receives information (A) from byte 0 of the bus interface 408 of the bi-endian processor 402.

One problem with the MIPS approach is that it is difficult to implement in existing computer systems. Specifically, the address controller for any target component must be redesigned. This requires additional modifications to the inner workings of an existing computer system.

Another problem not adequately solved by the MIPS approach concerns the situation where information is transmitted to some external medium from a component having a first byte ordering and subsequently transmitted from that medium to another component having a second byte ordering. This problem is illustrated below with regard to Figures 5 and 6.

Referring first to Figure 5, this Figure discloses examples of how bytes of a word of information in a memory device should appear after the information is received from a big endian processor. More specifically, Figure 5 shows the byte order of a word of information in a big endian memory 506 and a little endian memory 512 after the information is received from a big endian processor 502. Note that the byte order of the information is the same for both memory devices even though the "significance" of the bytes is

-11-

reversed (as discussed regarding Figure 3 above).

Also, note that the little endian memory 512 is able to receive the information from the big endian processor 502 only because an entire word of information is transmitted.

Referring now to Figure 6, the byte ordering shown on medium 602 results from medium 602 having received information from big endian processor 502 (via some I/O device, not shown) of Figure 5. As can be seen from the medium 602, byte 0 of big endian memory 506 (which, here, is representative of a byte address and contains information (D)) was written onto the medium 602 at location 0 and then bytes of increased significance were thereafter written to subsequent locations of medium 602. This represents the conventional way that information is written onto such media.

If the information on medium 602 were to be read back to a big endian memory 506, location 0 of the medium 602 would be received by byte 0 of big endian memory 506, location 1 by byte 1, etc. Thus, reading information from medium 602 back onto a big endian memory results in the information having the order shown by big endian memory 506.

If the information on medium 602 were transmitted to a little endian memory such as that shown by little endian memory 604, then, again, location 0 of medium 602 would be read to byte 0 of little endian memory 604, location 1 to byte 1, etc. When the entire word is read from the medium 602, the information would be ordered as shown by little endian memory 604. Note, however, that the order of the bytes is the reverse of what is shown by little endian memory 512 of Figure 5. Consequently, the byte information in little endian memory 604 is the reverse of what it should be.

The problem described above cannot be solved by manipulating the address scheme as per the MIPS solution, since the information is not lined up properly across the bus. In other words, the information would be incorrectly received by big endian processor 606 via bus 608 as shown, and manipulating the enablement of the byte addresses would not remedy the situation.

One solution to the above problem is to switch all of the bytes of information in little endian memory 604 so that they are aligned properly (that is, in this example, move information (D) to byte 3, move information (A) to byte 0, etc.) before allowing big endian processor 606 to access the information. This scheme is very inefficient, however, since it could take significant time to perform where large quantities of information (i.e., many words worth) are concerned.

It should be emphasized that the problem depicted by Figures 5 and 6 is significant, since some form of medium would likely be used to transmit information to a "mixed" computer system such as that depicted by little endian memory 604 and big endian processor 606. It should also be emphasized that the various examples described above are applicable not only to a processor/memory relationship, but almost any relationship between two components of a computer system.

#### SUMMARY OF THE INVENTION

The present invention overcomes the deficiencies of prior devices by providing a system and method for allowing components of different byte orderings to communicate information between each other in an effective manner, regardless of whether a single byte,

-13-

half word, word (or other multiples thereof depending upon bus widths) are transferred. The present invention achieves these results in an elegant manner by providing embodiments for crossing over byte lines of information in a communications link (e.g., a bus) between components when the two components have different byte orderings. Some embodiments indicate that upon receipt of an indication that the two components have the same byte ordering, the byte lines cease crossing over.

Some embodiments of the present invention contemplate use in existing systems, wherein one or more components having a different byte ordering than the rest of the computer system are inserted into the existing system. The present invention then allows these components of different byte ordering to communicate.

Embodiments of the present invention also contemplate computer systems (and uses within computer systems) having a bi-endian processor, so that operating systems (and other computer programs) having either a big endian or a little endian byte ordering can be used. In such a situation, the present invention crosses over the byte lines when the present invention is communicatively between the computer program running on the bi-endian processor and components having a different byte ordering from the computer program. Where the components are of the same byte ordering as the computer program, the present invention does not cross over the byte lines.

Some embodiments of the present invention contemplate that once the the present invention is set during operation of an operating system, it is not reset (i.e., made to switch modes from crossing over to not crossing over, or vice versa) during operation.

-14-

Other embodiments contemplate that the present invention can be dynamically reset depending upon the byte ordering of the computer program running at that time. In these latter embodiments, a computer program having a first byte ordering can call another computer program having a second byte ordering. That would necessitate the changing of the mode of the bi-endian processor as well as resetting whether the byte lines are crossed over or not.

#### BRIEF DESCRIPTION OF THE DRAWINGS

Various objects, features, and attendant advantages of the present invention can be more fully appreciated as the same become better understood with reference to the following detailed description of the present invention when considered in connection with the accompanying drawings, in which:

Figure 1 is a block diagram depicting the transfer of a single byte of information from a register to a component having a little endian byte ordering and to a component having a big endian byte ordering.

Figure 2 is a block diagram depicting the transfer of a half word of information from a register to a component having a little endian byte ordering and to a component having a big endian byte ordering.

Figure 3 is a block diagram depicting the transfer of a word of information from a register to a component having a little endian byte ordering and to a component having a big endian byte ordering.

Figure 4 is a block diagram depicting the transfer of information between two components having different byte orderings.

Figure 5 is a block diagram depicting the sequence of information written to a big endian memory and a little endian memory from a big endian processor.

Figure 6 is a block diagram depicting the sequence of bytes written from a medium to a little endian memory and thereafter transmitted to a big endian processor.

5        Figure 7A is a block diagram shows a conventional computer system using big endian components and a conventional computer system using little endian components.

10       Figure 7B is a block diagram of a computer system contemplated by embodiments of the present invention having a little endian processor and big endian components, and a computer system contemplated by embodiments of the present invention having big endian processor and little endian components.

15       Figure 8 is a block diagram of a computer system contemplated by embodiments of the present invention having a bi-endian processor with big endian components and a computer system contemplated by embodiments of the present invention having a bi-endian processor with  
20       little endian components.

Figure 9 is a diagram depicting embodiments of the present invention in an active mode.

25       Figure 10A is a diagram of a switch in an inactive mode as contemplated by embodiments of the present invention.

Figure 10B is a switch in an active mode as contemplated by embodiments of the present invention.

30       Figure 11 is a circuit diagram of a switch as contemplated by embodiments of the present invention.

Figure 12A is a diagram depicting the use of switches in an inactive mode as contemplated by embodiments of the present invention.

35       Figure 12B is a diagram depicting the use of switches in an active mode as contemplated by embodiments of the present invention.

Figure 13 is a flow diagram of a method contemplated by embodiments of the present invention for swapping information.

5 Figure 14 is an example of the operation of the present invention for transferring a single byte of information from a component having a first byte ordering to a component having a second byte ordering.

10 Figure 15 is an example of the operation of the present invention for transferring a half word of information from a component having a first byte ordering to a component having a second byte ordering.

15 Figure 16 is an example as contemplated by embodiments of the present invention for transferring information, originating from a component having a first byte ordering, from a medium to a component having a second byte ordering and then to a component having a first byte ordering.

20 Figure 17 is a block diagram of embodiments of the present invention for dynamically switching between utilizing a computer program having a first byte ordering to utilizing a computer program having a second byte ordering.

25 Figure 18 is a flow diagram of a method contemplated by embodiments of the present invention for dynamically changing the byte ordering of the processor to accommodate computer programs of varying byte ordering.

### 30 DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

#### I. Overview and Contemplated Configurations

35 This invention relates to a system and method for allowing a computer-related component having a first byte ordering to effectively communicate with a computer-related component having a second byte

ordering. Embodiments of the present invention also envision a system and method for determining whether two components have the same byte ordering and then for making provisions to allow the two components to communicate with each other. In addition, embodiments of the present invention contemplate that these provisions allowing the two components to communicate with each other can be administered dynamically.

In general, the present invention utilizes an elegant scheme for overcoming the deficiencies of the prior devices. Specifically, the present invention contemplates an information "swapping" scheme for allowing components having a first byte ordering to communicate with components having a second byte ordering. Even with all the intricacies discussed in the Background section above concerning byte management within the two byte ordering schemes and the complexities of facilitating interaction therebetween, the swapping scheme contemplated by the present invention achieves the desired results.

Before discussing the details of schemes envisioned for use in implementing the actual swapping of information, various configurations contemplated by embodiments of the present invention in which swapping occurs and environments contemplated for use with some of these embodiments are now discussed. These embodiments contemplate use in existing computer systems as well as (in whole or in part) themselves comprising entire computer systems. These embodiments are now described with regard to Figures 7A-7C.

Referring first to Figure 7A, this Figure depicts two conventional computer systems, each having a "master" and two "slave" components. Specifically, these computer systems each comprise a processor (the "master"), a memory and an I/O device (the "slaves").

-18-

More specifically, a computer system 702A comprises a big endian processor (e.g., a Motorola 68000) with a big endian memory and a big endian I/O. Similarly, a computer system 704A comprises a little endian processor (e.g., an Intel 80486), a little endian memory and a little endian I/O. Only operating systems and other computer programs designed to function (e.g., that were compiled) in a big endian environment can run on computer system 702A. Similarly only operating systems and other computer programs designed to function in a little endian environment can run on computer system 704A.

In order to allow the conventional computer systems of Figure 7A to run software of a different byte ordering from that which they were originally intended to run, their processors would have to operate using the same byte ordering of the operating system. That is, for example, if it were desired to make a big endian computer system use a little endian operating system, the processor would need to somehow run in a little endian mode. One way to do this is to simply exchange the processor in the computer system for a processor of opposite byte ordering (that is, the byte ordering of the operating system). This alone, however, is insufficient, as the problem discussed in conjunction with Figure 4 in the Background section would then exist. A solution to this problem is now discussed with regard to Figure 7B.

Referring now to Figure 7B, computer system 702B contains a big endian memory 712 and a big endian I/O 714, just as does computer system 702A of Figure 7A. However, in computer system 702B, a little endian processor 706 has replaced the big endian processor of computer system 702A. Also, a swapper 708 is communicatively positioned between the little endian

processor 706 and the other components of computer system 702B, including a bus 710. This swapper 708 allows information to be effectively communicated between the little endian processor 706 and big endian components such as big endian memory 712. This, in effect, enables the little endian operating system and computer programs running on little endian processor 706 to communicate with big endian memory 712 and big endian I/O 714.

The swapper 708 accomplishes the above-mentioned feat by "swapping" (i.e., switching around) bytes in a manner that will be described further below. It is noted here, however, the byte addresses of the various components are not modified and that the information is swapped by the swapper 708 as the information travels to its destination.

Similarly to computer system 702B, in computer system 704B a big endian processor 716 has replaced the little endian processor of computer system 704A. Again, the presence of swapper 708 enables the big endian operating system and computer programs running on big endian processor 716 to communicate with a little endian memory 722 and a little endian I/O 724.

Embodiments of the present invention contemplate that the replacement the processor with one having the opposite byte ordering would likely require some modification of the processor interface into the computer system. Thus, some modification of the "glue logic" would be required. The implementation of such modifications would be apparent to one skilled in the art.

In addition to placing the swapper 708 in communication with a processor and other components as shown in the computer systems of Figure 7B, embodiments of the present invention also contemplate that the

swapper 708 can be communicatively positioned between components in ways other than shown by Figure 7B. For example, referring to computer system 702A, the big  
5        endian memory can be replaced with a swapper 708 and a little endian memory. Consequently, any component(s) having a second byte ordering can be introduced into a computer system having components of a first byte ordering and can function in that computer system by using the swapper 708.

10        Embodiments of the present invention also contemplate computer systems having a parallel processing architecture where each processor is in communication with a swapper 708.

15        Rather than replace the processor of a computer system, some embodiments of the present invention (and environments used thereby) contemplate the use of a bi-endian processor such as the MIPS R-4000 discussed  
20        above. Since the R-4000 can operate in either a big endian mode or a little endian mode, an operating system (and any other computer program) of either big endian or little endian byte ordering can be utilized by this processor.

25        Since the vast majority of components (e.g., memory and I/O) are designed for either big endian or little endian byte ordering (but not both), it is typically most efficient to design a computer system utilizing a bi-endian processor with components all having big  
30        endian byte ordering or little endian byte ordering. Though the present invention contemplates embodiments where this is not the case, the examples of Figure 8 disclose such situations.

35        Referring now to Figure 8, a computer system 802 is shown with a bi-endian processor 806, big endian memory 712, big endian I/O 714 and bus 710. A swapper 708 is in communication with bi-endian processor 806 and the

-21-

5 other components of the computer system 802. If a big  
endian operating system is running on the bi-endian  
processor 806 (which itself would be configured in big  
endian mode) then the swapper 708 would be inactive  
(i.e., no byte swapping would occur) since all  
components in the computer system 802 have a big endian  
byte ordering. However, if a little endian operating  
system were running on bi-endian processor 806 (and,  
thus, the bi-endian processor 806 is configured in a  
10 little endian mode) then the swapper 708 would be  
active in order to allow information to effectively be  
used when passed between the little endian operating  
system and the big endian components of computer system  
802. The same concept applies to computer system 804,  
15 where little endian components are used rather than big  
endian components.

With regard to the embodiments shown in Figure 8,  
some of these embodiments contemplate that the bi-  
endian processor 806 is set in accordance with the byte  
ordering of a particular operating system and is not  
20 re-set during operation. Other embodiments of the  
present invention contemplate that the bi-endian  
processor 806 can be re-set dynamically during  
appropriate circumstances, such as where a big endian  
operation system calls a little endian software routine  
25 (or vice versa). In that situation, the swapper 708  
would change its state (i.e., from active to inactive,  
or vice versa) at that time as well. Using the example  
of computer system 804, the swapper 708 would change  
30 from inactive to active if a big endian operating  
system called a little endian software routine.

Embodiments of the present invention contemplate  
that memory components of either byte ordering can be  
any type of computer memory device including DRAM's or  
CMOS. Some embodiments contemplate that the memory is  
35

controllable by such schemes as direct memory access. Embodiments of the present invention envision that the I/O components are interfaces (such as hard disk controllers or local area network controllers) to I/O devices.

Embodiments of the present invention also contemplate that the swapper 708 can be an integral part of one or more components (such as a bus or memory device) of a computer system rather than a separate entity.

## II. Contemplated Implementations

Embodiments of the present invention envisioned to implement the swapping of information to facilitate the features discussed above are now described. First, the swapper 708, as envisioned by various embodiments of the present invention, is conceptually described in conjunction with Figure 9.

Referring now to Figure 9, four lines (A, B, C and D) are shown passing through swapper 708 and crossing therein. Each of these lines is one byte in width. Although Figure 9 discloses four lines (hereafter referred to as "byte lines") any number of byte lines are envisioned for use by various embodiments of the present invention.

From Figure 9, it can be seen that the outer-most byte lines (specifically, byte lines A and D) swap positions with each other. The inner-most byte lines (B and C) also swap positions with each other. If more byte lines were used, a similar pattern would emerge. For example, if two more byte lines existed, one between byte lines A and B and the other between C and D, these two byte lines would also swap position through swapper 708.

The swapper 708 as shown in figure 9 is displayed in an "active" mode (i.e., the byte lines are swapped). As indicated above, the swapper 708 can also be set to an "inactive" mode, in which the byte lines are not swapped (i.e., it were as though there were no swapper 708 at all). Embodiments for implementing the swapper 708 are discussed below.

Some of the embodiments envisioned by the present invention for implementing the swapper 708 utilize one or more "switches." Each of these switches connectively crosses over (i.e., swaps) two byte lines when the swapper is set to an "active" mode. An example of these switches can be seen from Figures 10A and 10B. The labeling of the byte lines for these Figures (as well as in Figures 11, 12A and 12B below) is related, for clarity, to the labeling of the byte lines in Figure 9.

Referring first to Figure 10A, a switch 1002 is shown having byte lines A and D passing through it. When a non-cross over signal (e.g., a "0", as shown in Figure 10A, and indicative of "inactive mode" of the swapper 708) is transmitted to a cross-over input 1004, then switch 1002 is "inactive" and byte lines A and D are not swapped. If, however, a cross-over signal (e.g., a "1") is transmitted to cross-over input 1004, then switch 1002 is "active" and byte lines A and D are swapped as shown by Figure 10B. Of course, the embodiments of the present invention contemplate that values other than those discussed in conjunction with Figures 10A and 10B can be used to indicate that switch 1002 is active or inactive.

More detailed embodiments of switch 1002 are now discussed in conjunction with an example shown by Figure 11. Referring now to Figure 11, switch 1002 comprises 4 switches 1104 implemented as shown. These

switches 1104 can be, for example, transistors. Also, an inverter 1102 is implemented as shown. In this particular example, when cross-over input 1004 is "low" (e.g., as represented by a digital "0") byte lines 1 and 2 are not swapped. If, however, cross-over input 1004 is "high" (e.g., as represented by a digital "1") then byte lines 1 and 2 are swapped.

It should be understood that the present invention contemplates various configurations for implementing switch 1002 other than those disclosed in conjunction with Figure 11.

An example of how the switch 1002 is used in swapper 708 as contemplated by embodiments of the present invention is now discussed with regard to Figures 12A and 12B. Referring first to Figure 12A, two switches (1002A and 1002B) are utilized in swapper 708. Since each switch controls 2 byte lines, swapper 708 thus provides for control of four bytes (i.e., 32 bits) of information (as also indicated by Figure 9 above).

As shown by Figure 12A, byte line A comprises the 8 least significant bits (that is, bits 0-7) and byte line D comprises the 8 most significant bits (that is, bits 24-31). These two byte lines share the same switch 1002A within swapper 708. In other words, it is these two byte lines that are swapped when a cross-over signal is received by cross-over input 1004. The remaining two byte lines that pass through swapper 708 (byte lines B and C) comprise bits 8-15 and bits 16-23, respectively, and share switch 1002B. It is contemplated that byte lines A-D on each side of swapper 708 are in communication with some component (not shown) each having a big endian or a little endian byte ordering.

In the example of Figure 12A, a non-cross over signal (in this case, a "0") is shown as being received by cross-over input 1004. Consequently, the byte lines are not swapped. Figure 12A thus represents the situation when both components on either side of swapper 708 have the same byte ordering. That is, both components with which the byte lines of swapper 708 are in communication have either a big endian or a little endian byte ordering.

Conversely, Figure 12B, represents the situation when the components on either side of the swapper 708 do not have the same byte ordering. This is indicated by the fact that a cross-over input signal (in this case, a "1") is received by cross-over input 1004. Thus, switches 1002A and 1002B are "active" and swap byte lines as shown. Specifically, byte lines A and D are swapped and byte lines B and C are swapped. This has the effect discussed in conjunction with Figure 9 above.

It should be noted that the cross-over input signal, in effect, acts as an indication that the source component (i.e., the component transmitting the information) has a byte ordering different from the target component.

Embodiments of the present invention contemplate that the combination of the two switches 1002A and 1002B comprising the swapper 708 of Figures 12A and 12B can be implemented using a product called "Quick Switch" (part number 74QST3383) made by Quality Semiconductor of Santa Clara, CA. However, it should be understood that the present invention also contemplates other schemes for implementing swapper 708, such as by the use of ASIC technology. These other schemes may or may not utilize "switches" (e.g., they might utilize muxes).

5 The present invention contemplates that the cross-over input signal can be generated as a result of various stimuli in order to notify the swapper 708 when the byte lines should be swapped. An example might be for the processor to write to some port address to which cross-over input 1004 is responsive. Whether or not the swapper 708 should be made active could be set interactively by a user or by storing an active/non-active setting in some non-volatile memory. In this way, the swapper 708 can be appropriately set based upon the byte ordering status of the components of the computer system.

10 Although Figures 9, 12A and 12B show the swapper 708 handling 4 bytes of information, it should be understood that this is only by way of example and that the present invention contemplates embodiments for swapping any number of bytes. Also, the present invention contemplates embodiments where the swapper 708 is permanently "active" where it is known that the components between which it communicatively resides will always be of a different byte ordering.

15 Embodiments of a method of operation contemplated by the present invention are now discussed with regard to Figure 13. Referring to Figure 13, the first step is to determine the byte ordering used by a first component within a computer system. This is indicated by a block 1302. Then, a determination is made of the byte ordering used by a second component (with which the first component communicates). This is indicated by a block 1304.

25 A determination is then made concerning whether the byte ordering used by the first component is the same as that for the second component. This is indicated by a decision block 1306. If they are the same, then no further action needs to be taken, and communications

30

35

-27-

between the two components can immediately commence. This is indicated by a block 1310 in conjunction with decision block 1306.

If, however, the byte ordering used by the first component is not the same as for the second component, then the swapper 708 must be activated. This is indicated by a block 1308. Once activated, communications between the first and second components may then commence, as indicated by block 1310.

### III. Illustrations Of Operation

Examples demonstrating the effective operation of embodiments of the present invention are explained with regard to Figure 14. Referring now to Figure 14, an example depicting a computer system having a big endian processor 1402 and a little endian memory 1406 is disclosed. In this example, information is transmitted via a four byte bus 1410 between the big endian processor 1402 and little endian memory 1406. The swapper 708 is shown in an "active" state (since the byte ordering of the two components is different) and is connected to bus 1410 between big endian processor 1402 and little endian memory 1406.

In this specific example, a register 1408 contains a single byte of information (information (A)). When transmitted onto bus 1410 via a bus interface 1404, information (A) is transmitted at byte 0 as shown. If there were no intervening swapper 708, information (A) would be received at byte 3 of little endian memory 1406 (which is not enabled to receive information) as described in Figure 4 in the Background section above. This is the result of standard protocols used with memory management schemes of computer systems contemplated for use in (and in environments used with) embodiments of the present invention. Because of these

protocols, when information is transmitted at byte 0 of the transmitting component (in this case, big endian processor 1402) then byte 0 of the receiving component (in this case, little endian memory 1406) is enabled.

5           Thus, in the present invention, the active swapper 708 "swaps" information (A) so that it is delivered to the appropriate location. Specifically, Information (A) is received by the byte address of the little endian memory (that is, byte 0) that is enabled.

10           Figure 15 shows an example where a half-word (i.e., two bytes) of information is transmitted rather than the single byte discussed in conjunction with the previous Figure. Referring now to Figure 15, information (A) and (B) are contemplated to reside in register 1408 as shown. This information is further contemplated to be transmitted onto bus 1410 via bus interface 1404, such that information (A) is transmitted at byte 0 and information (B) is transmitted at byte 1. In view of the memory management schemes mentioned above, writing this information to little endian memory 1406 causes bytes 0 and 1 of little endian memory 1406 to become enabled. As can be seen by Figure 15, the swapper 708 then allows the information to be received by the enabled portions of little endian memory 1406.

20           It should be understood that the present invention is not limited to the examples of Figures 14 and 15. For example, bus widths of any number of bytes are contemplated and the concepts discussed above apply regardless of the number of bytes transmitted or to what byte locations (i.e., byte addresses) in the target component they are transmitted to. Also, although these examples show a big endian "master" component (in this case, a processor) in communication with a little endian slave component (in this case,

35

memory) it should be understood that the present invention contemplates other combinations, including slave-to-slave and master-to-master combinations.

5 The present invention also overcomes the problem described in Figures 5 and 6 in the Background section concerning receipt of information from a medium. An example of this is now described with regard to Figure 16.

10 Referring to Figure 16, a medium 1602 (which can be any type of transportable medium such as a floppy disk, tape or even a local area network) is shown having received 4 bytes of information from a big endian component as discussed with regard to Figures 5 and 6 above. Thus, information (D), which was transmitted  
15 from byte 0 of the big endian component, is shown at location 0 of medium 1602. Information (C) is shown at location 1, etc.

20 When information is received by little endian memory 1406 from medium 1602 (via some I/O device, not shown), the byte locations are matched up. In other words, whatever information is at location 0 of medium 1602 is placed at byte 0 of little endian memory 1406; whatever is at location 1 is placed at byte 1, etc. This is due to industry standard protocol as discussed  
25 above.

30 When the information is placed into little endian memory 1406 as discussed above, the byte order of the information is the same as shown by the little endian memory 604 of Figure 6. As discussed in conjunction with that Figure, the information in little endian memory 604 cannot, as is, be used by big endian processor 606. To be useful, the information needs to be in the format it was originally generated in (which is the format shown by big endian processor 502 and big  
35 endian memory 506 in Figure 5).

Referring back to Figure 16, the swapper 708, as contemplated by embodiments of the present invention, allows big endian processor 1402 to receive and use the information from the little endian memory 1406.

5 Specifically, swapper 708 swaps the information so that, for example, information (D) (which is transmitted at byte 0 from little endian memory 1406) is received at byte 0 of the big endian processor 1402 rather than at byte 3 as occurred in the example of  
10 Figure 6. Thus, the information received by big endian processor 1402 as a result of the swapper 708 is in the format which the information was originally transmitted by the big endian processor 502 of Figure 5. In this way, it is not necessary to "swap" the information  
15 while it resides in memory, thus saving considerable time.

Embodiments of the present invention contemplate that an I/O component (not shown) is used to facilitate transmission of information between medium 1602 and  
20 little endian memory 1406. In general, I/O components contemplated for use by embodiments of the present invention as described above envision that information is transmitted (as between other components) to and from the I/O components in a byte-wide fashion. If  
25 information is transmitted using parallel methods (i.e., transmitted using more than one byte width) then the information needs to be swapped prior to transmitting or receiving information to or from the I/O component.

30 Some embodiments of the present invention contemplate that taking care of parallel methods can be done using a device driver which swaps the data prior to writing it to (or reading it from) the I/O component. Design and implementation of such a device  
35 driver would be known to one skilled in the art. Also,

an additional swapper 708 can instead be placed in communication between the I/O component and the rest of the computer system.

#### IV. Dynamic Swapper Embodiments

As indicated above, the byte ordering mode of a bi-endian processor used in a computer system envisioned by various embodiments of the present invention can be changed dynamically so that, in conjunction with the swapper 708, software routines having a byte ordering different from the byte ordering of the operating system can be used. An example of such a scheme is shown by Figure 17.

Referring now to Figure 17, a bi-endian processor 1708 is contemplated to be initially configured in a first byte ordering mode (i.e., either big endian or little endian) and is operating using an operating system (not shown) having a first byte ordering. A computer program 1704 within a memory 1702, both of which have the first byte ordering, can consequently be accessed and run by the bi-endian processor 1708 without the need to use swapper 708 (i.e., swapper 708 is "inactive," and the byte lines are not swapped).

If, within the computer program 1704 there exists a call to a software routine 1706 (the software routine 1706 being initially generated using a processor having a second byte ordering), then two things happen. First, the bi-endian processor 1708 switches its byte ordering mode to the second byte ordering mode to accommodate this software routine 1706. Second, since the bi-endian processor 1708 is now using a different byte ordering than memory 1702, the swapper 708 becomes active (i.e., it is reset to an active mode). Consequently, the byte lines within the swapper 708 will swap the bytes of information as discussed above.

Once the bi-endian processor 1708 and swapper 708 have been set to accommodate the software routine 1706, software routine 1706 then is received by the bi-endian processor 1708. Upon completion of execution (or if the software routine 1706 itself called another software routine having a first byte ordering) the bi-endian processor 1708 and swapper 708 revert back to their initial state.

Embodiments of the present invention contemplate that instructions exist within computer program 1702 in conjunction with the call to software routine 1706 to set the bi-endian processor 1708 and swapper 708 to a different state. However, the present invention also contemplates embodiments where the state can be set in other ways.

Figure 18 depicts embodiments of a method of operation of the present invention. Referring now to Figure 18, the first step is that computer program 1704 is executed in bi-endian processor 1708 running in a first byte ordering mode. This is indicated by a block 1802. The computer program 1704 then calls a software routine 1706 configured to execute in a second byte ordering mode. This is indicated by a block 1804.

The next step is to reconfigure the bi-endian processor 1708 and swapper 708 to run in a second byte ordering mode. That is, the byte ordering of the bi-endian processor 1708 is switched and the swapper 708 is made active. This is indicated by a block 1806. Once this is done, the software routine 1706 is loaded and executed in the bi-endian processor 1708, as indicated by a block 1808.

Although the embodiments of the present invention shown by Figures 17 and 18 contemplate that the computer program 1704 and memory 1702 have the same byte ordering while software routine 1706 has a

different byte ordering, it should be understood that embodiments of the present invention also contemplate that the software routine 1706 can have the same byte ordering as memory 1702, while the computer program 1704 has a different byte ordering from memory 1702. In that case, the swapper 708 would be active during the execution of computer program 1704 and inactive during the execution of software 1706.

It should be understood that the present invention also contemplates use in other situations and examples where it is necessary or advantageous to dynamically change the active/inactive status of the swapper 708.

It should be emphasized that the various components of embodiments of the present invention can be implemented in hardware, software or a combination thereof. In such embodiments, the various components and steps would be implemented in hardware and/or software to perform the functions of the present invention. Any presently available or future developed computer software language and/or hardware components can be employed in such embodiments of the present invention.

It should also be understood that the present invention is not limited to the embodiments indicated above, and that the examples presented above are merely for the purposes of illustration. The scope of the present invention should therefore be interpreted by the following claims as defined by the forgoing figures and text.

What is claimed is:

1. An apparatus for facilitating the transmission of information in a computer system, wherein the information is comprised of bytes, comprising:

a first component having a first byte ordering, said first component capable of providing information to the computer system;

a second component, said second component capable of receiving the information from said first component; and

swapping means, communicatively positioned between said first component and said second component, for swapping the bytes of the information, prior to its receipt by said second component when the byte ordering of said second component is a second byte ordering.

2. The apparatus of claim 1, wherein said first component is a bi-endian processor, initially configured using a first byte ordering mode,

wherein said bi-endian processor is running a first computer program having said first byte ordering.

3. The apparatus of claim 2, wherein upon receipt of a request by said first computer program to run a second computer program having a second byte ordering, said bi-endian processor is reconfigured using a second byte ordering,

wherein said swapping means ceases swapping the bytes of information where the bytes of information are swapped when said bi-endian processor is running said first computer program and wherein said swapping means swaps the bytes of information where the bytes of information are not swapped when said bi-endian processor is running said first computer program.

-35-

4. The apparatus of claim 1, wherein said swapping means comprises at least one switch means within which two byte lines pass through,

5 wherein said at least one switch means connectively swaps two byte lines upon receipt of a cross over input signal indicating that said switch is to become active.

10 5. The appartaus of claim 1, wherein said swapping means is an integral part of said first component or said second component.

15 6. An apparatus for facilitating the transmission of information in a computer system, wherein the information is comprised of bytes and is transmitted by a source component having a byte ordering, comprising:

a target component for receiving the information, wherein said target component has a first byte ordering; and

20 swapping means, in communication with said target component, for receiving an indication concerning the byte ordering of the source component;

25 wherein upon receiving an indication that the source component has a byte ordering different from said first byte ordering of said target component, said swapping means is set to an active mode in which the bytes of the information are swapped prior to the information being received by said target component,

30 wherein upon receiving an indication that the source component has said first byte ordering, said swapping means is set to an inactive mode in which the bytes of the information are not swapped prior to the information being received by said target component.

7. The apparatus of claim 6, wherein said target component is a memory device.

5 8. The apparatus of claim 6, wherein the source component is a bi-endian processor, initially configured using a first byte ordering mode, wherein said bi-endian processor is running a first computer program having said first byte ordering.

10 9. The apparatus of claim 8, wherein upon receipt of a request by said first computer program to run a second computer program having a second byte ordering, said bi-endian processor is reconfigured using a second byte ordering,

15 wherein said swapping means is reset to an active mode when said swapping means is set to an inactive mode and wherein said swapping means is reset to an inactive mode when said swapping means is set to an active mode.

20 10. A computer system for utilizing a computer program comprised of bytes, wherein the computer program has a byte ordering, comprising:

25 processor means for operating in either a first byte ordering mode or a second byte ordering mode, wherein said processor means is configured to operate using the byte ordering mode of the computer program;

30 a component, having a byte ordering, for storing the computer program and for allowing said processor means to access the computer program;

35 swapping means, communicatively positioned between said processor means and said component, for receiving an indication of the byte ordering mode of said processing means, and for swapping the bytes of the

computer program when the byte ordering of said component is different from the byte ordering of said processor means.

5 11. The apparatus of claim 10, wherein said swapping means comprises at least one switch means within which two byte lines pass through,

10 wherein said at least one switch means connectively swaps two byte lines upon receipt of a cross over input signal indicating that said switch is to become active.

15 12. The apparatus of claim 10, wherein said component is a memory device.

20 13. A method for facilitating the transmission of information between a first component and a second component in a computer system, wherein the first component and the second component each have a byte ordering, comprising the steps of:

(1) determining whether the byte ordering of the first component is the same as the byte ordering of the second component;

25 (2) activating a byte swapper upon determination in said step (3) that the byte ordering of the first component is not the same as the byte ordering of the second component; and

30 (3) deactivating said byte swapper upon determination that the byte ordering of the first component is the same as the byte ordering of the second component.

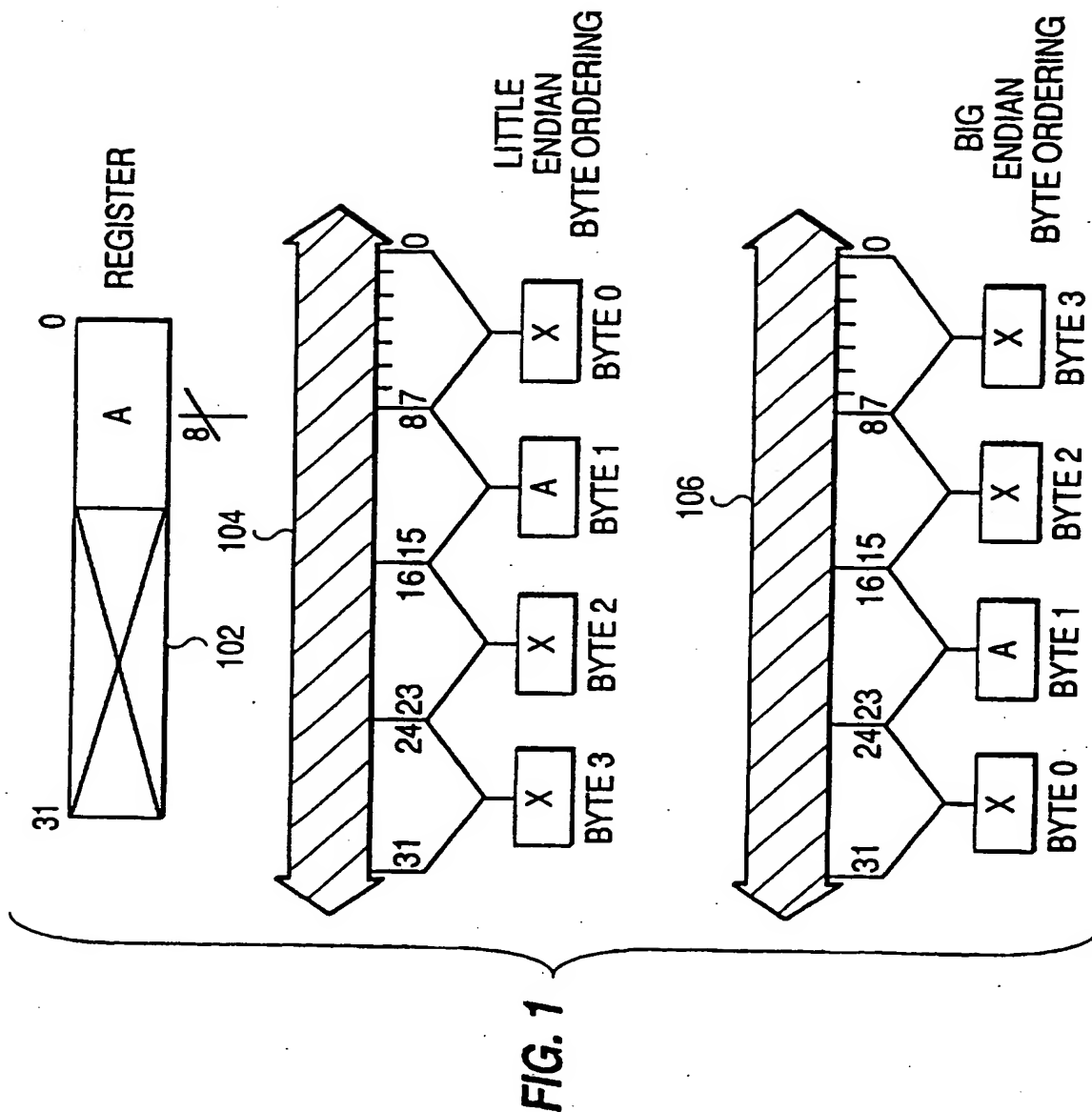
35 14. The method of claim 13, further comprising the steps prior to said step (1) of:

-38-

receiving information concerning the byte ordering  
of the first component; and

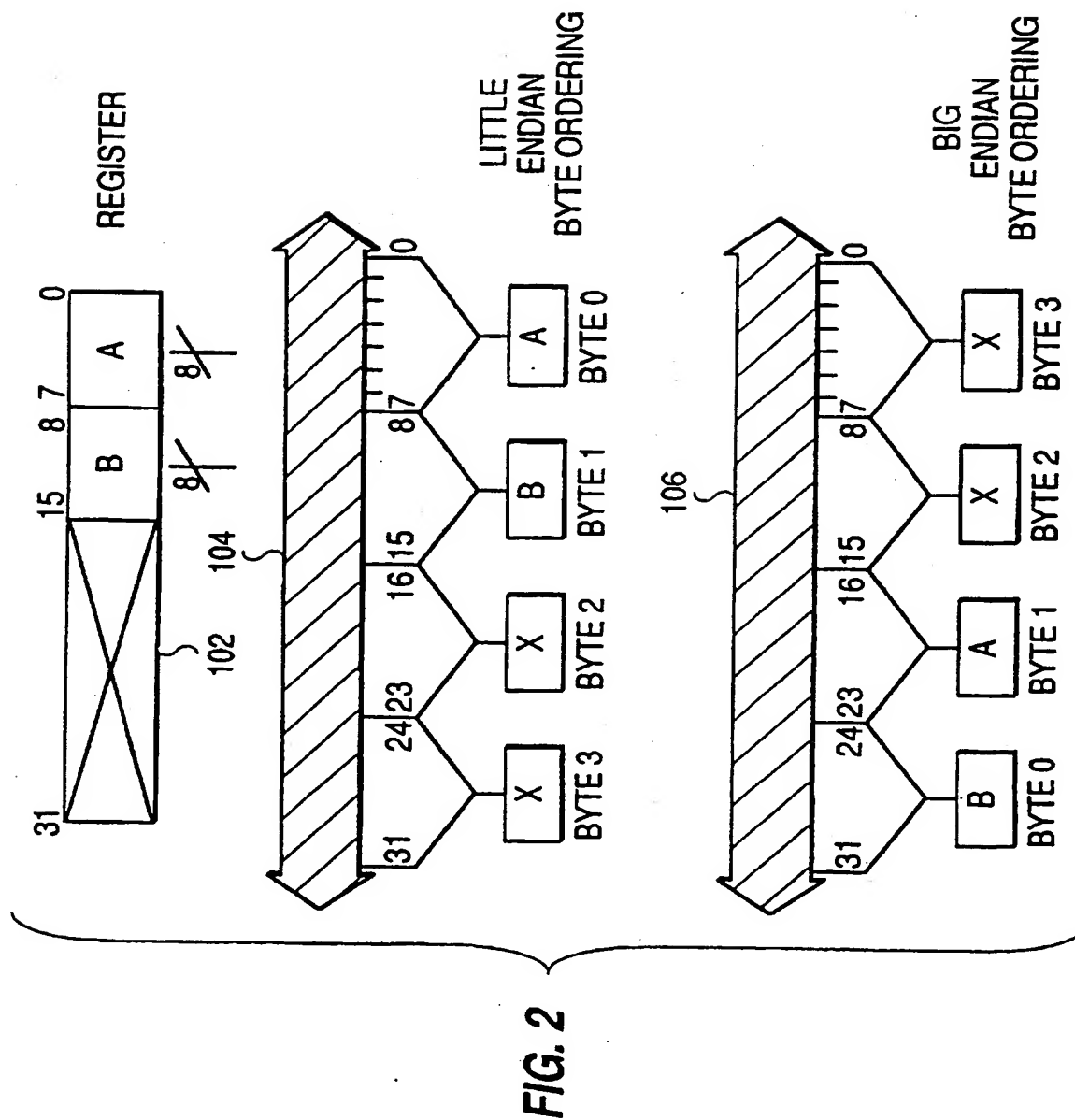
receiving information concerning the byte ordering  
of the second component.

1/17

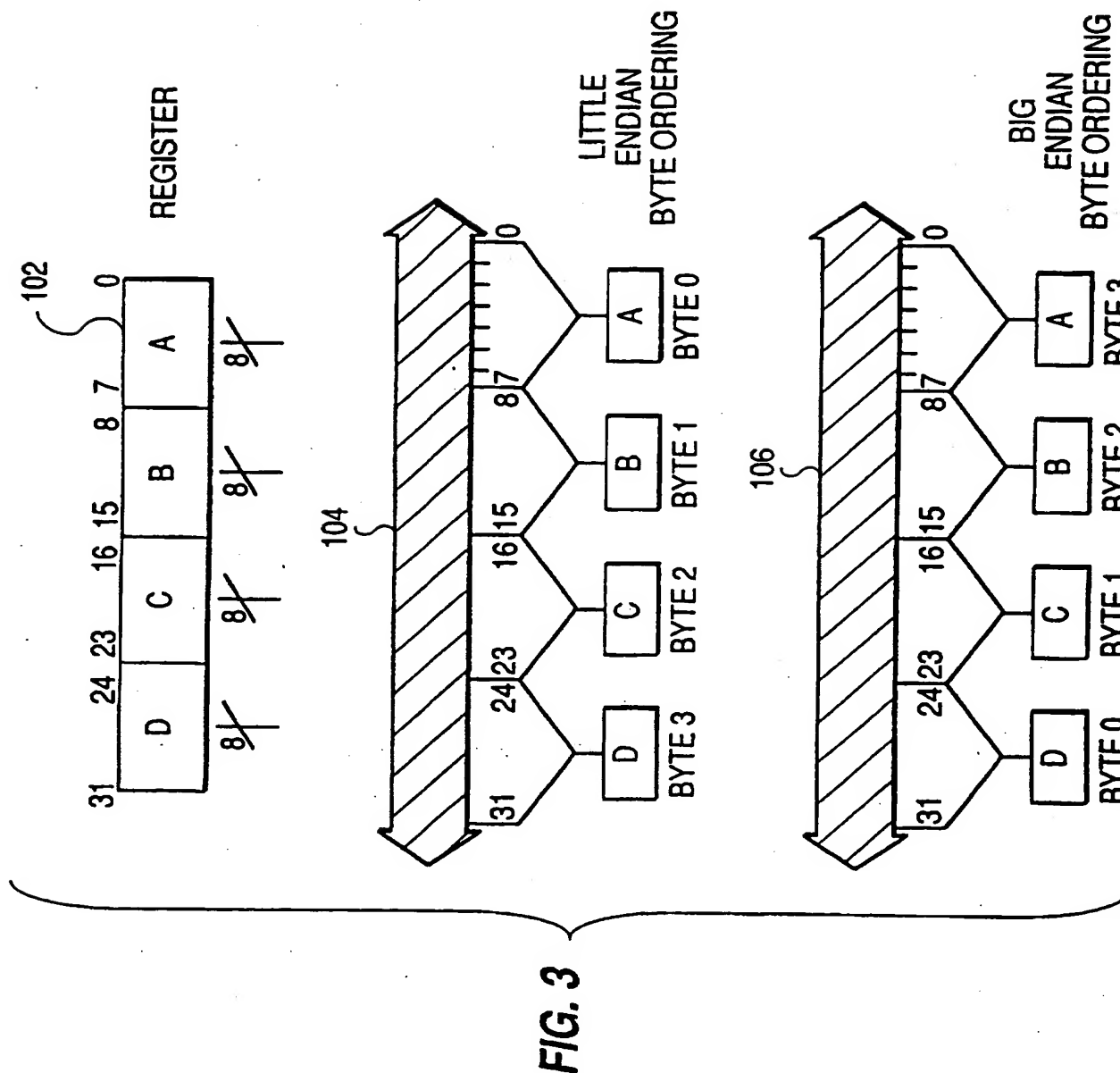


SUBSTITUTE SHEET (RULE 26)

2/17

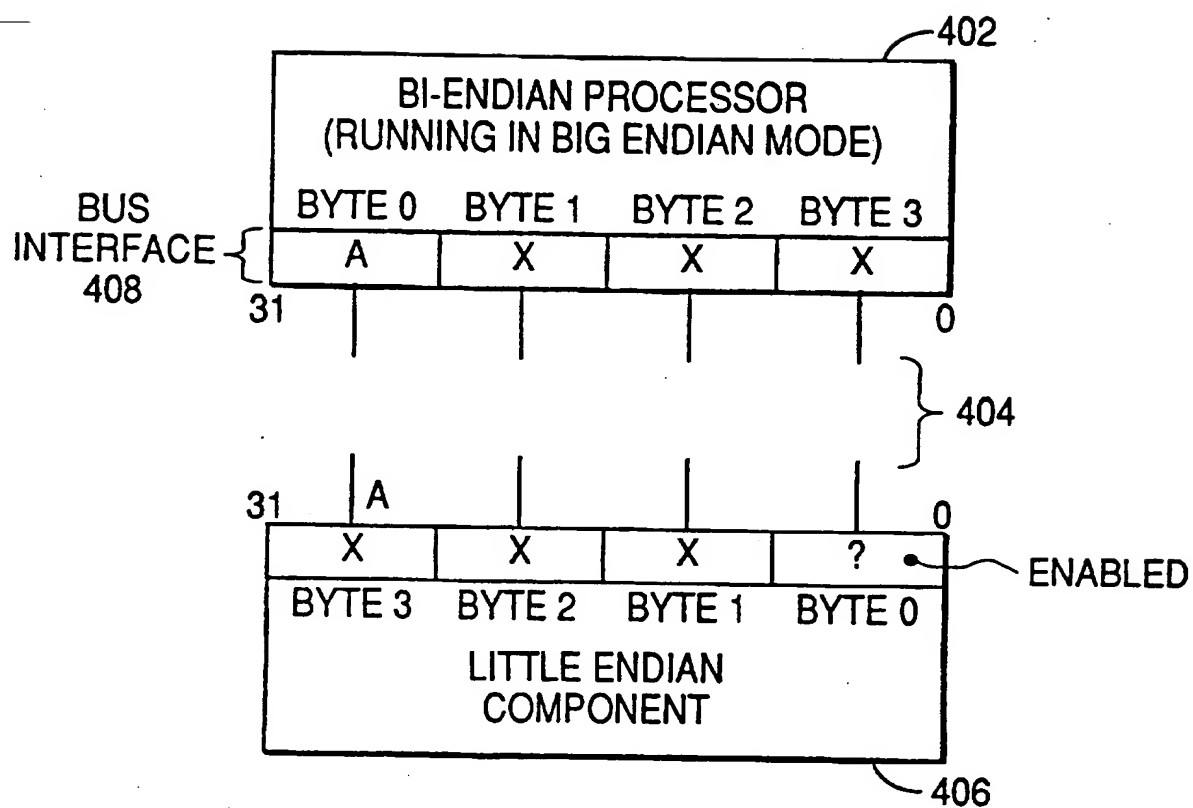


3/17



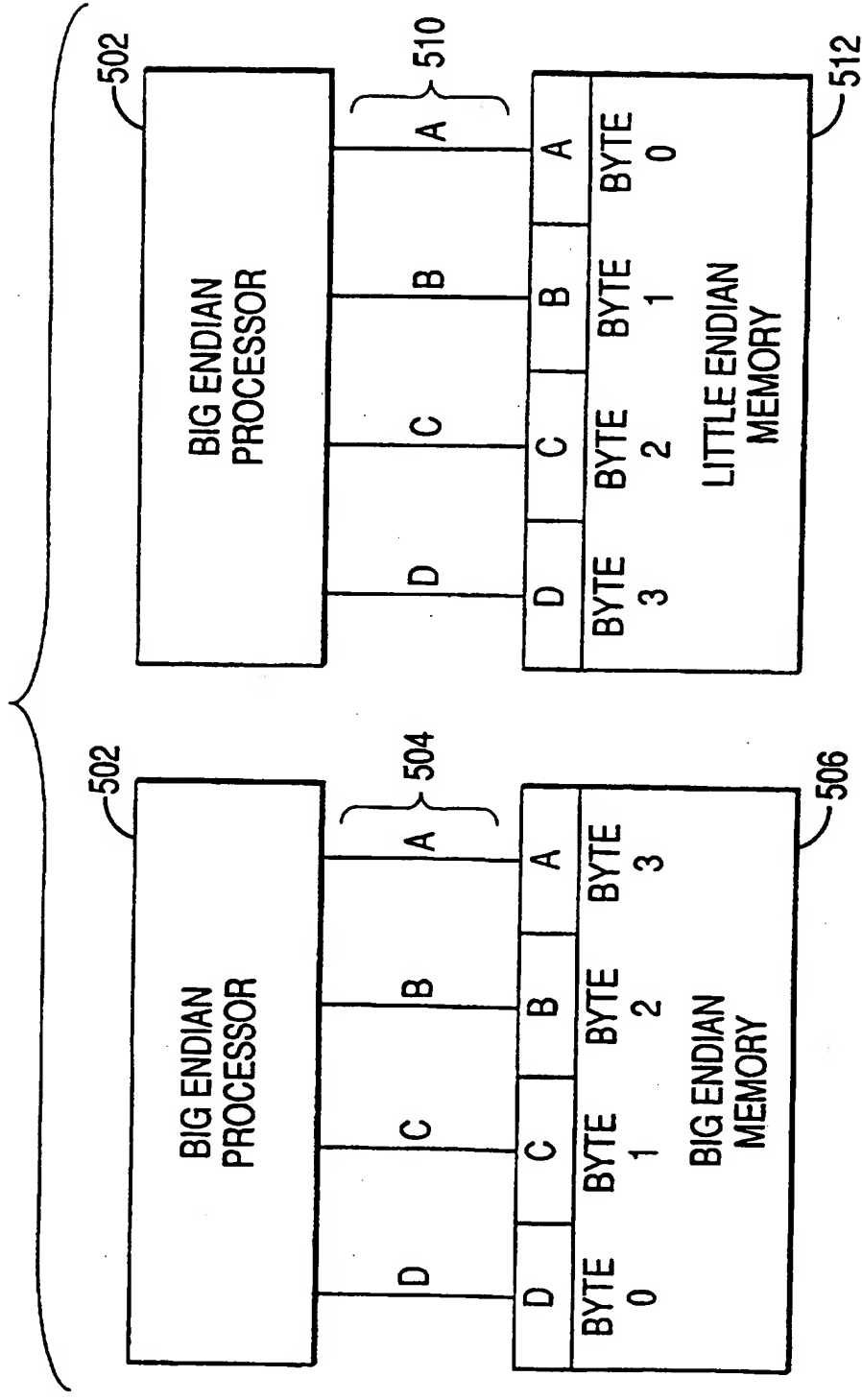
4/17

FIG. 4

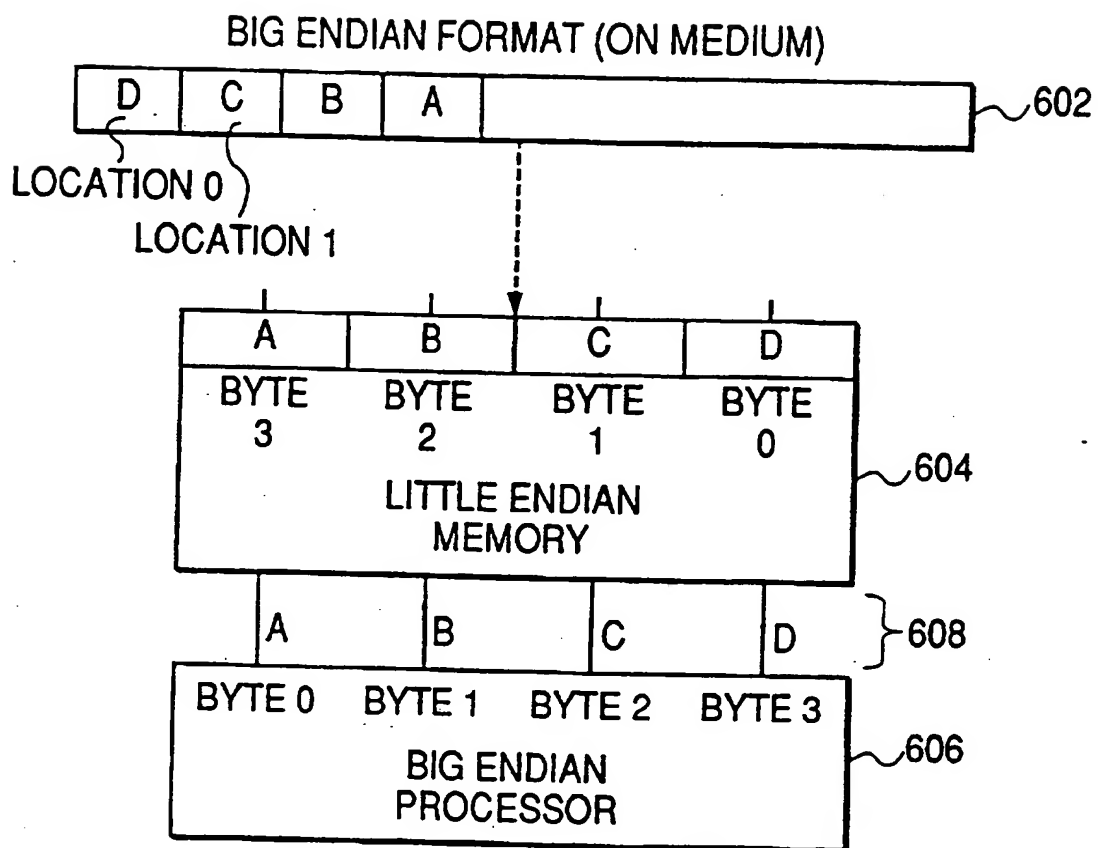


5/17

FIG. 5



6/17

**FIG. 6**

SUBSTITUTE SHEET (RULE 26)

7/17

FIG. 7A

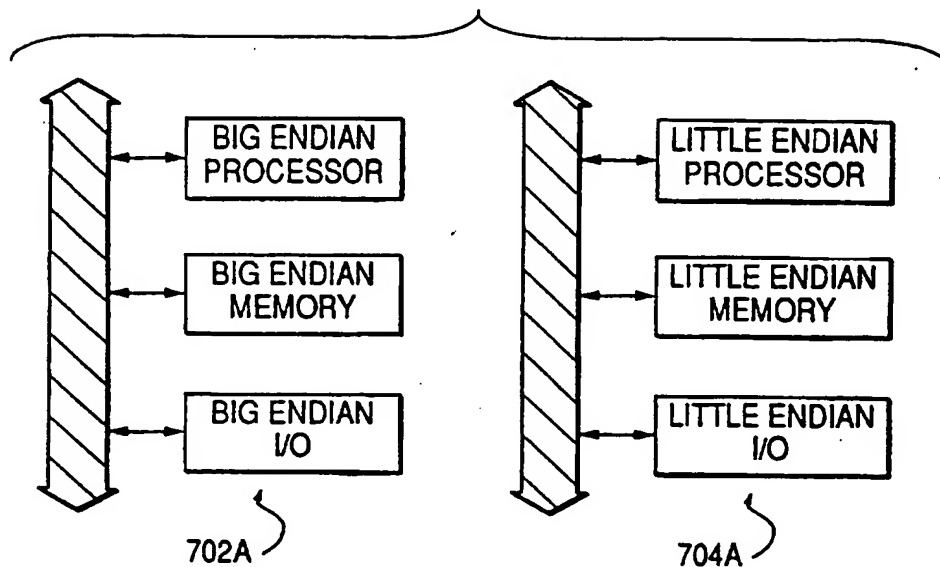
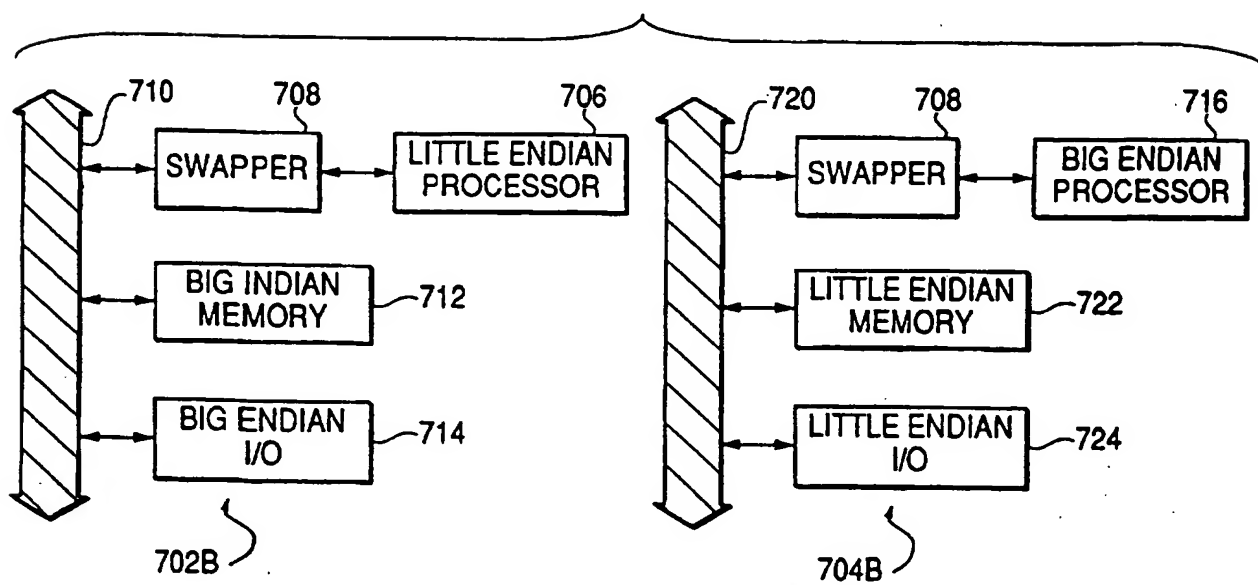


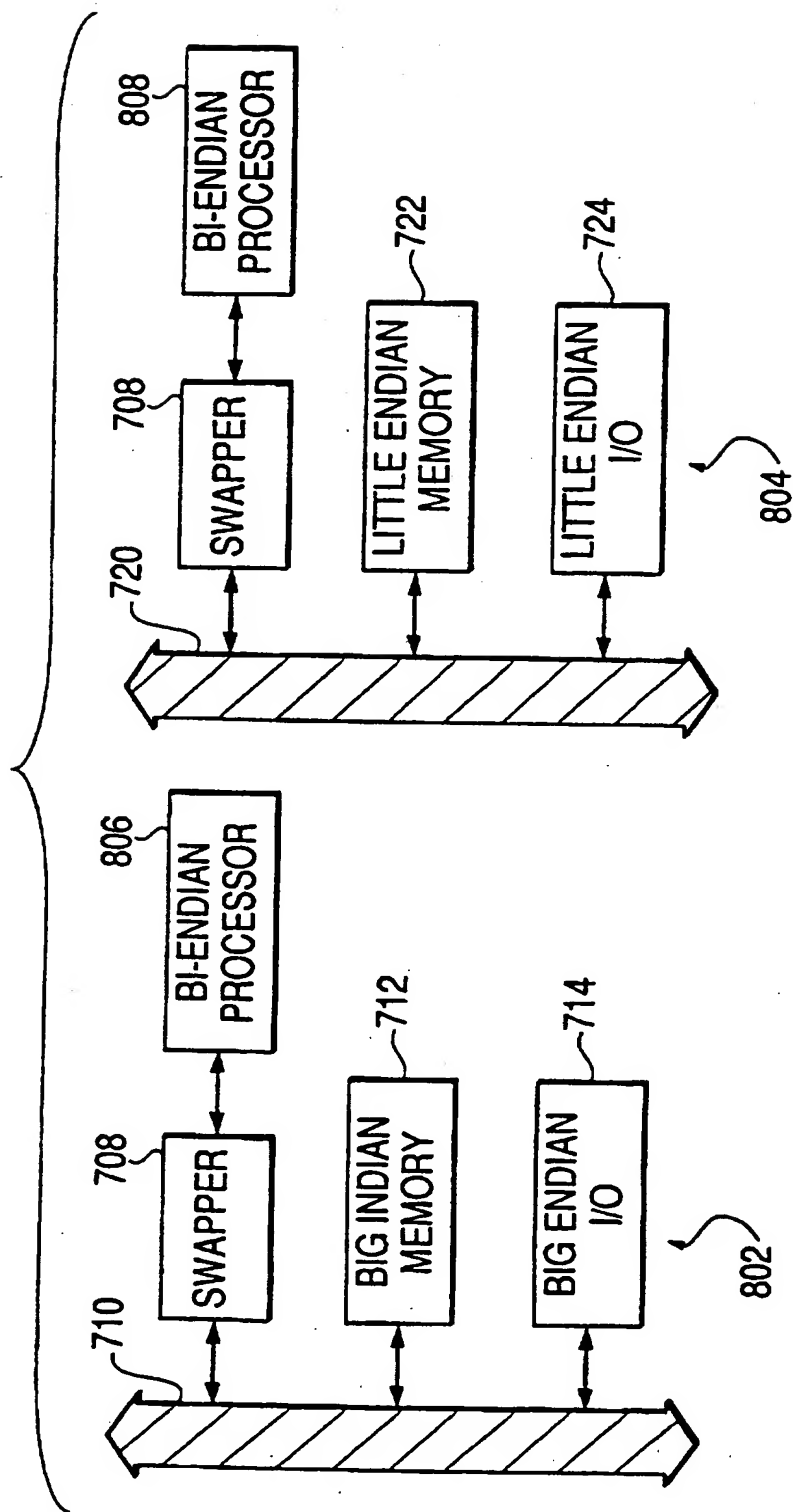
FIG. 7B



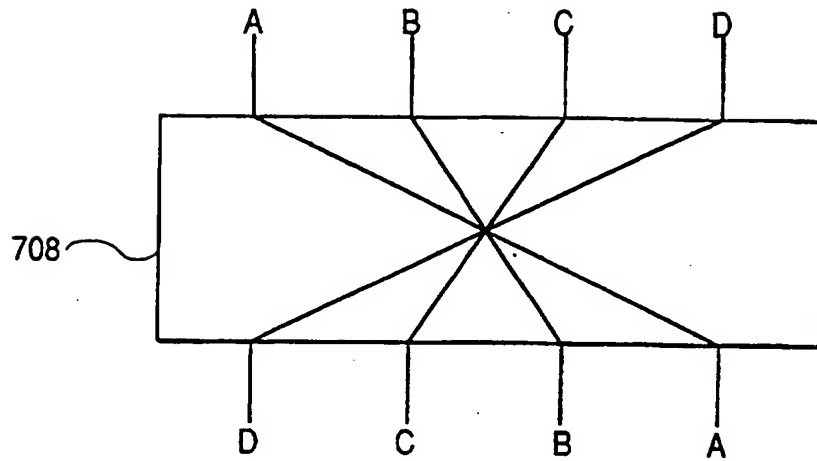
SUBSTITUTE SHEET (RULE 26)

8/17

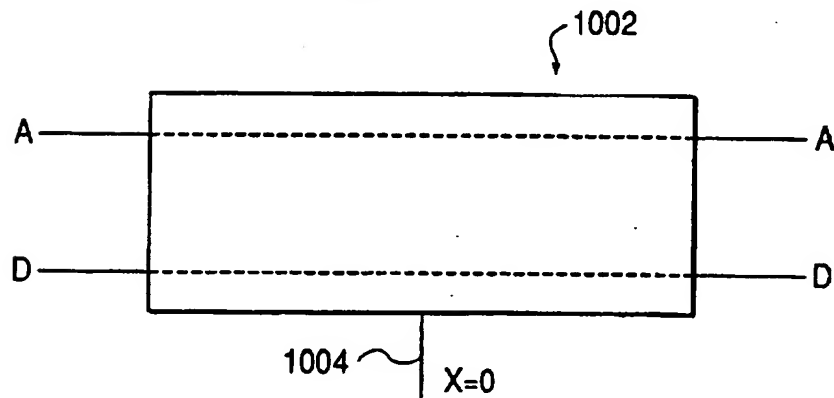
FIG. 8



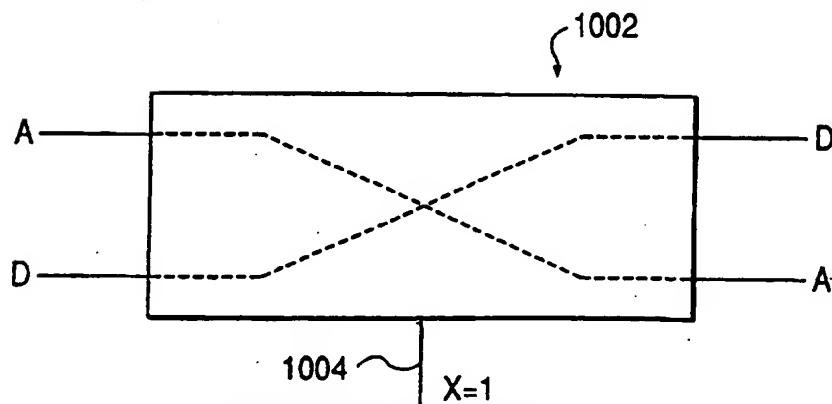
9/17  
**FIG. 9**



**FIG. 10A**



**FIG. 10B**



SUBSTITUTE SHEET (RULE 26)



II/I7

FIG. 12A

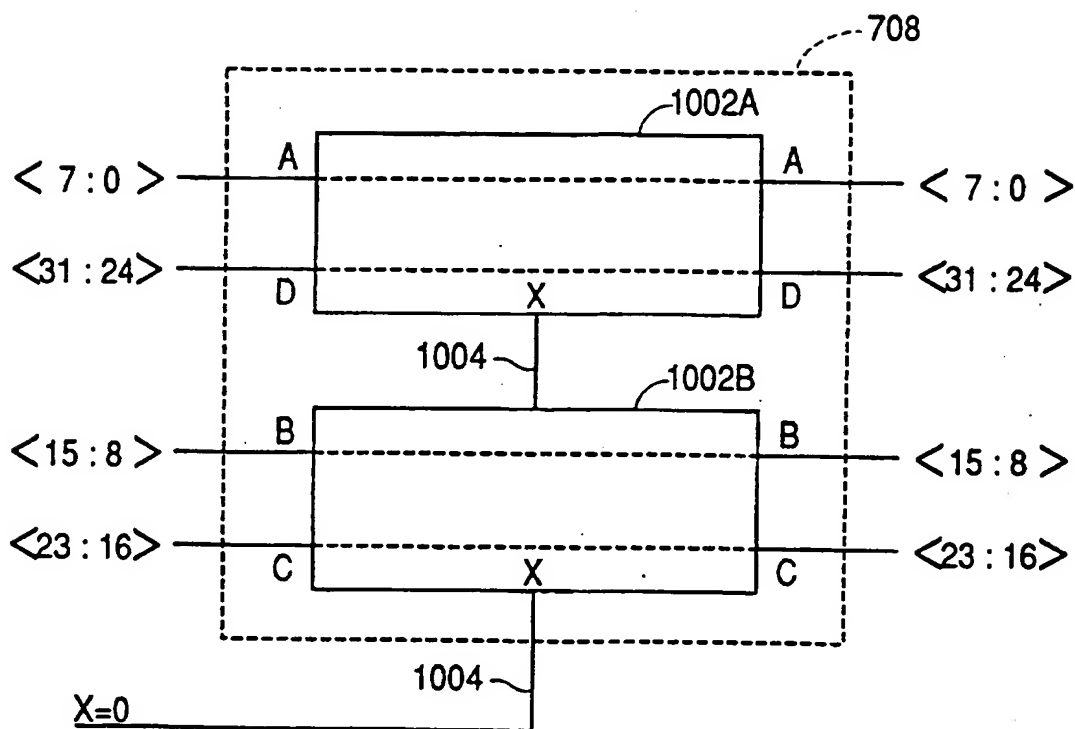
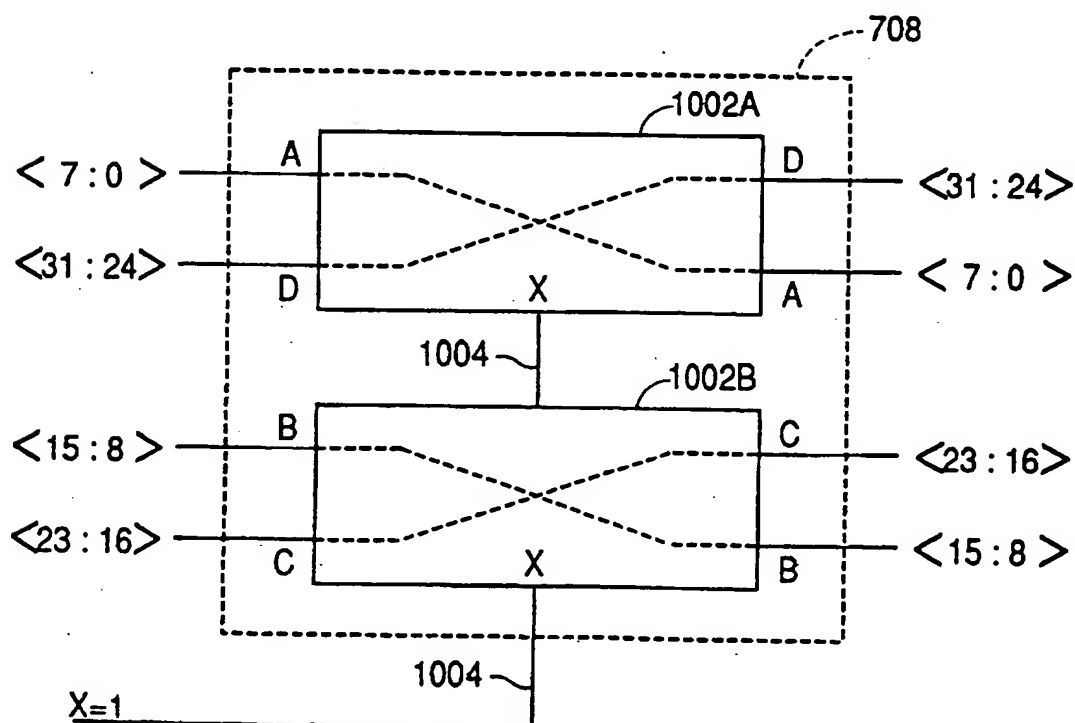
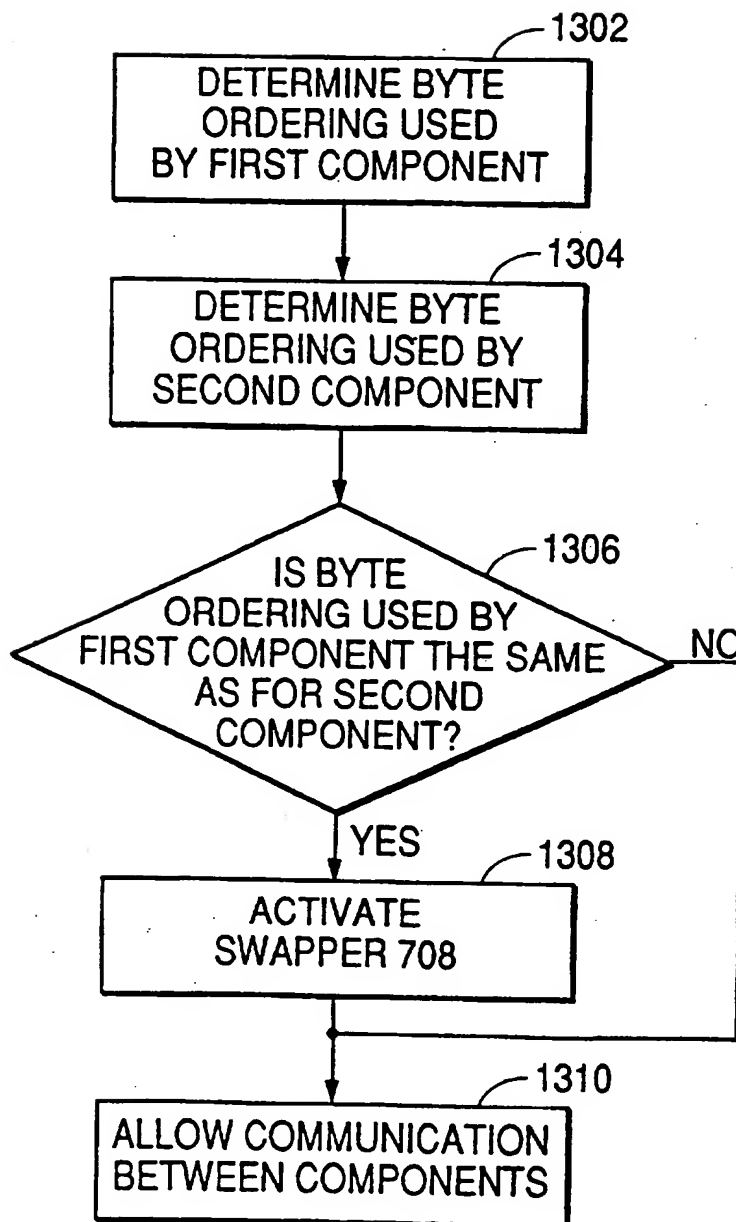


FIG. 12B



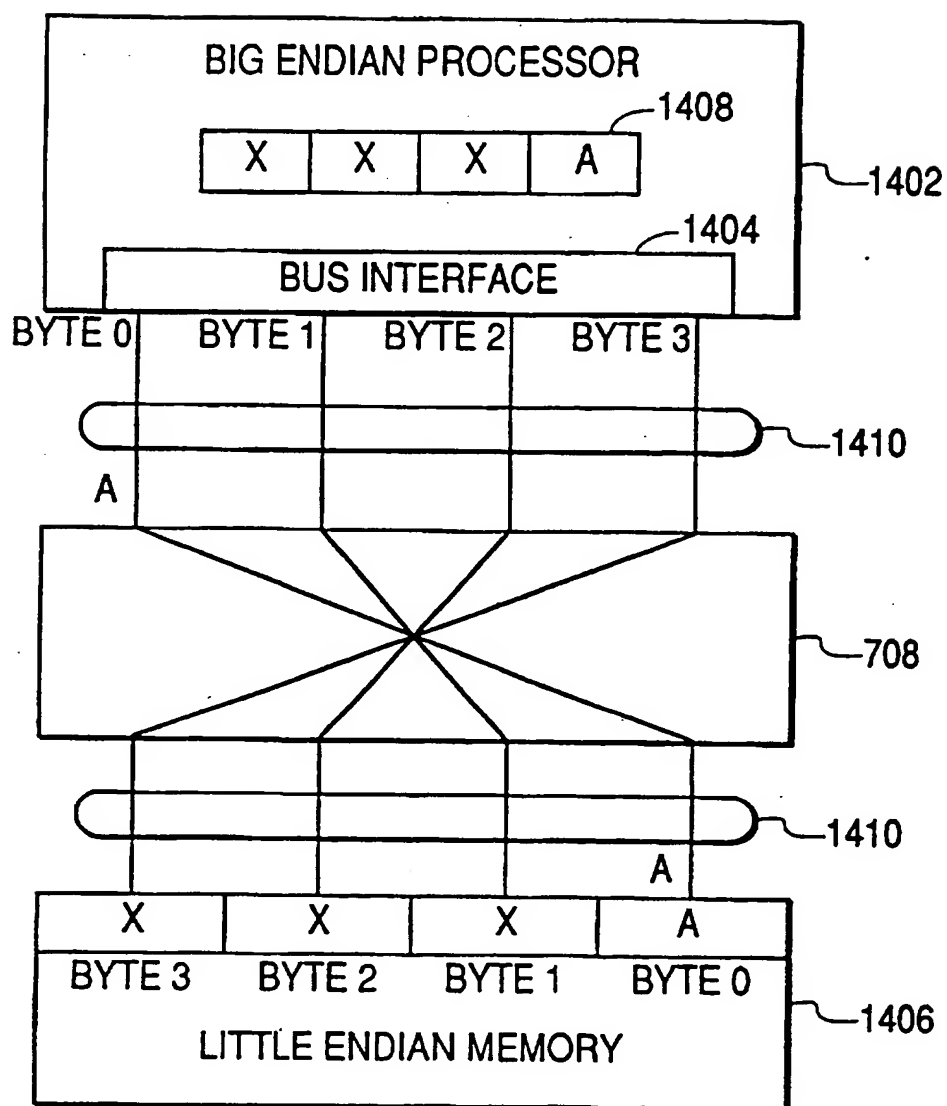
SUBSTITUTE SHEET (RULE 26)

12/17

**FIG. 13**

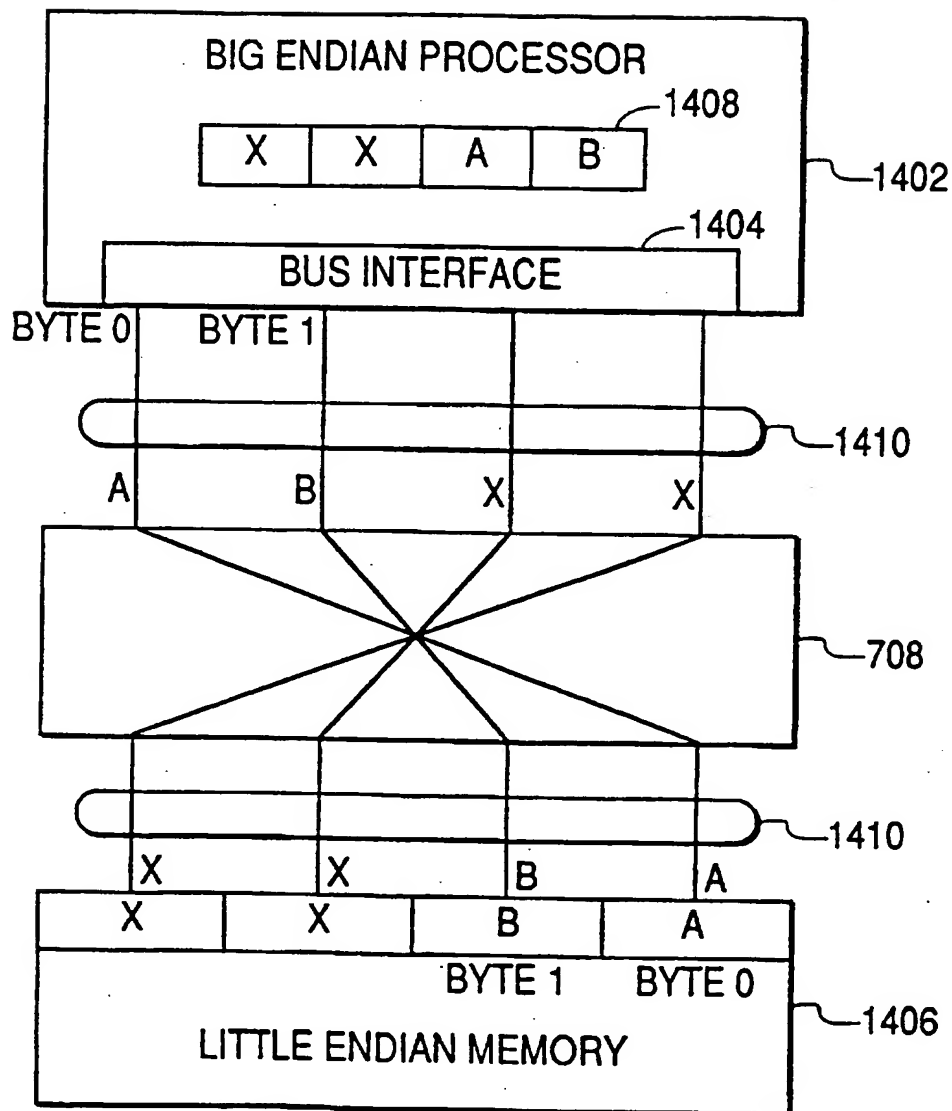
SUBSTITUTE SHEET (RULE 26)

13/17

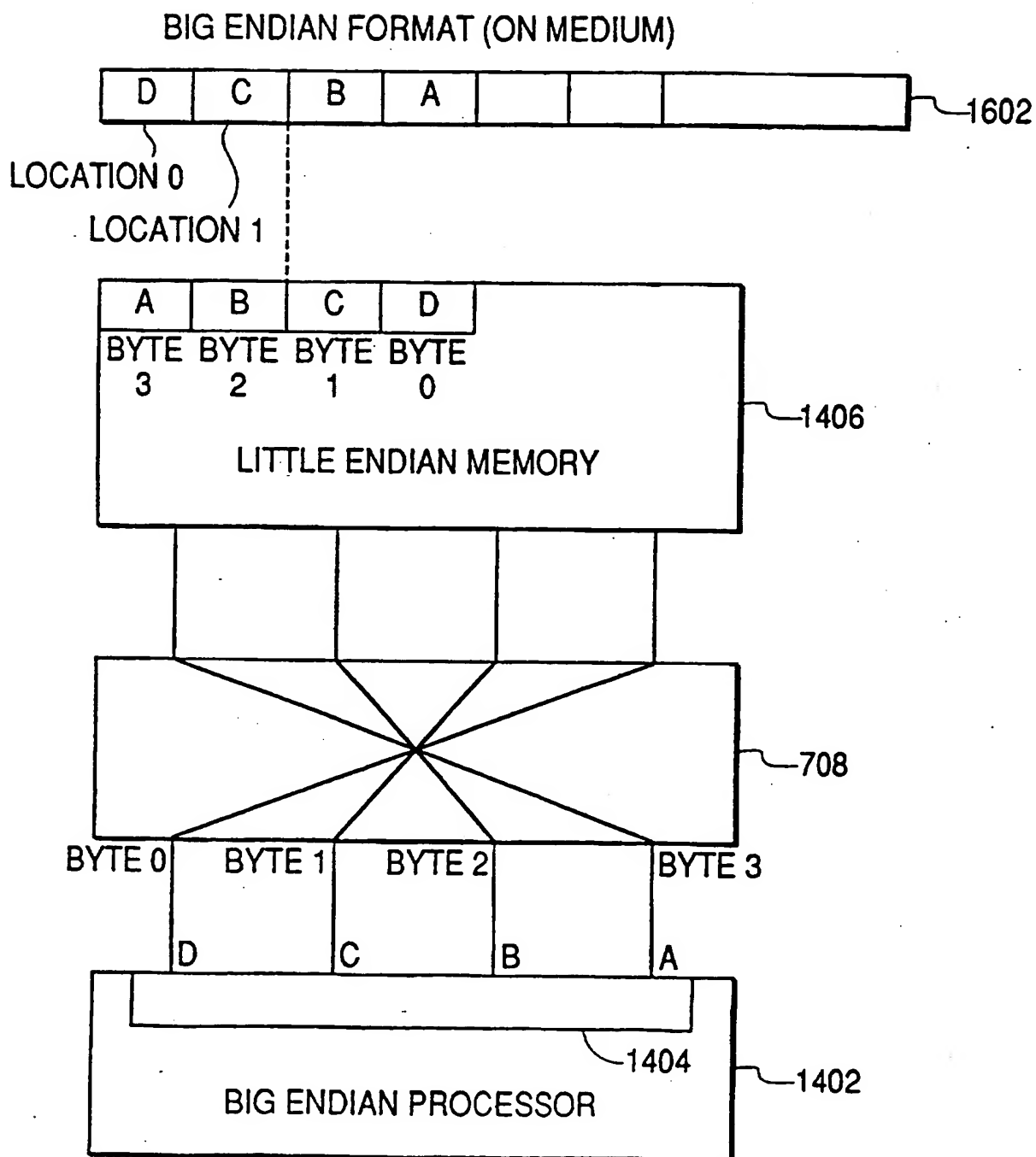
**FIG. 14**

SUBSTITUTE SHEET (RULE 26)

14/17

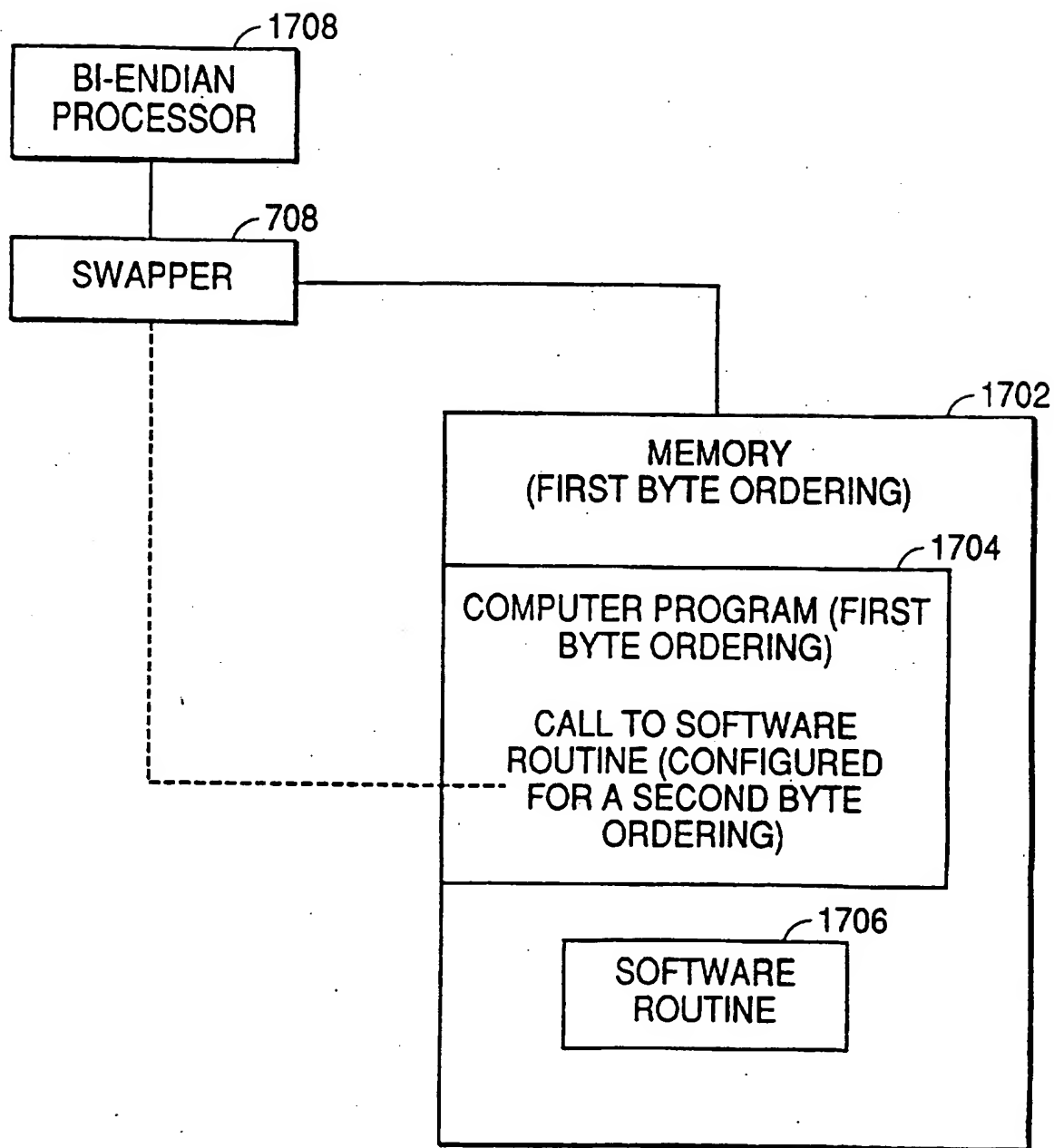
**FIG. 15**

15/17

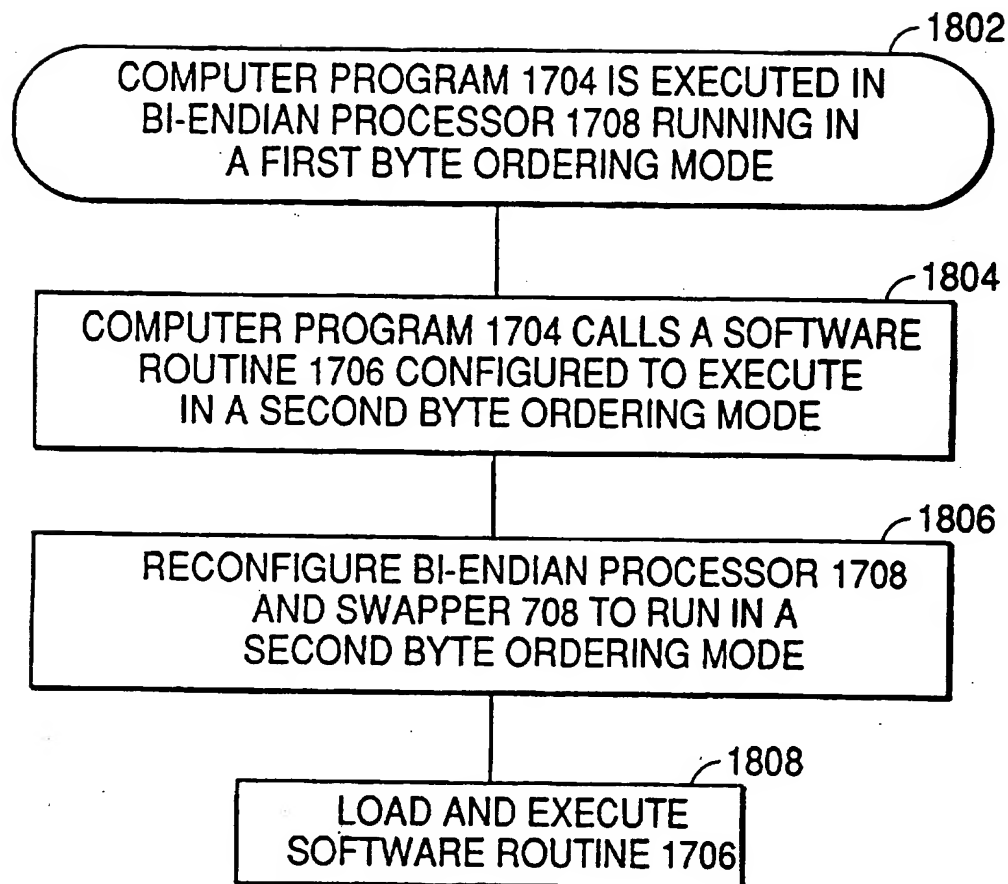
**FIG. 16**

SUBSTITUTE SHEET (RULE 26)

16/17

**FIG. 17****SUBSTITUTE SHEET (RULE 26)**

17/17

**FIG. 18**

SUBSTITUTE SHEET (RULE 26)

**THIS PAGE BLANK (USPTO)**

PCT

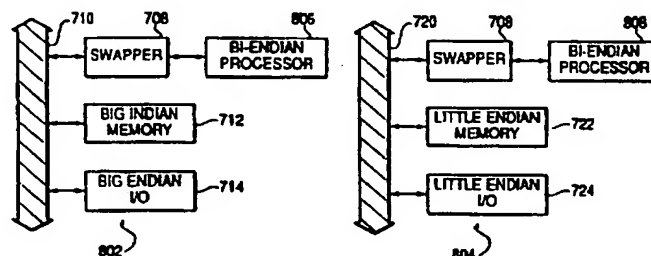
WORLD INTELLECTUAL PROPERTY ORGANIZATION  
International Bureau



INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification <sup>5</sup> : G06F 7/00, 13/40	A3	(11) International Publication Number: WO 94/15269 (43) International Publication Date: 7 July 1994 (07.07.94)
(21) International Application Number: PCT/US93/12416 (22) International Filing Date: 17 December 1993 (17.12.93) (30) Priority Data: 07/994,405 21 December 1992 (21.12.92) US (71) Applicant: OLIVETTI ADVANCED TECHNOLOGY CENTER, INC. [US/US]; 20300 Stevens Creek Boulevard, Cupertino, CA 95014 (US). (72) Inventor: TAM, Stephen, Yiu-Ming; 2858 Dominici Drive, Fremont, CA 94536 (US). (74) Agents: McKIE, Edward, F., Jr. et al.; Banner, Birch, McKie & Beckett, 1001 G Street, N.W., 11th floor, Washington, DC 20001-4597 (US).		(81) Designated States: JP, European patent (AT, BE, CH, DE, DK, ES, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).  Published <i>With international search report. Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.</i>  (88) Date of publication of the international search report: 24 November 1994 (24.11.94)

(54) Title: APPARATUS, SYSTEM AND METHOD FOR FACILITATING COMMUNICATION BETWEEN COMPONENTS HAVING DIFFERENT BYTE ORDERINGS



(57) Abstract

A system and method for allowing a component having a first byte ordering to effectively transfer information to another component having a second byte ordering. The present invention envisions embodiments where facilitation of the information transmission is set depending upon whether the two components have the same byte ordering or whether they have different byte orderings.

**THIS PAGE BLANK (USPTO)**

**FOR THE PURPOSES OF INFORMATION ONLY**

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AT	Austria	GB	United Kingdom	MR	Mauritania
AU	Australia	GE	Georgia	MW	Malawi
BB	Barbados	GN	Guinea	NE	Niger
BE	Belgium	GR	Greece	NL	Netherlands
BF	Burkina Faso	HU	Hungary	NO	Norway
BG	Bulgaria	IE	Ireland	NZ	New Zealand
BJ	Benin	IT	Italy	PL	Poland
BR	Brazil	JP	Japan	PT	Portugal
BY	Belarus	KE	Kenya	RO	Romania
CA	Canada	KG	Kyrgyzstan	RU	Russian Federation
CF	Central African Republic	KP	Democratic People's Republic of Korea	SD	Sudan
CG	Congo	KR	Republic of Korea	SE	Sweden
CH	Switzerland	KZ	Kazakhstan	SI	Slovenia
CI	Côte d'Ivoire	LI	Liechtenstein	SK	Slovakia
CM	Cameroon	LK	Sri Lanka	SN	Senegal
CN	China	LU	Luxembourg	TD	Chad
CS	Czechoslovakia	LV	Latvia	TG	Togo
CZ	Czech Republic	MC	Monaco	TJ	Tajikistan
DE	Germany	MD	Republic of Moldova	TT	Trinidad and Tobago
DK	Denmark	MG	Madagascar	UA	Ukraine
ES	Spain	ML	Mali	US	United States of America
FI	Finland	MN	Mongolia	UZ	Uzbekistan
FR	France			VN	Viet Nam
GA	Gabon				

# INTERNATIONAL SEARCH REPORT

Inter national Application No  
PCT/US 93/12416

A. CLASSIFICATION OF SUBJECT MATTER  
IPC 5 G06F7/00 G06F13/40

According to International Patent Classification (IPC) or to both national classification and IPC

## B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

IPC 5 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

## C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US,A,5 107 415 (K. SATO ET AL.) 21 April 1992 see the whole document ---	1-14
X	EP,A,0 282 969 (HITACHI) 21 September 1988 see the whole document ---	1-14
A	IEEE MICRO, vol.3, no.4, August 1983, NEW YORK US pages 32 - 47 H. KIRRMANN 'Data Format and Bus Compatibility in Multiprocessors' see the whole document ---	1-14
A	EP,A,0 470 570 (MIPS) 12 February 1992 cited in the application see abstract see column 3, line 24 - line 45 ---	1,2,6-8, 10,12
-/--		

☒ Further documents are listed in the continuation of box C.

☒ Patent family members are listed in annex.

### \* Special categories of cited documents :

- \*A\* document defining the general state of the art which is not considered to be of particular relevance
- \*E\* earlier document but published on or after the international filing date
- \*L\* document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
- \*O\* document referring to an oral disclosure, use, exhibition or other means
- \*P\* document published prior to the international filing date but later than the priority date claimed

- \*T\* later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
- \*X\* document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
- \*Y\* document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.
- \*A\* document member of the same patent family

Date of the actual completion of the international search

6 October 1994

Date of mailing of the international search report

19.10.94

Name and mailing address of the ISA

European Patent Office, P.B. 5818 Patentlaan 2  
NL - 2280 HV Rijswijk  
Tel. (+ 31-70) 340-2040, Tx. 31 651 epo nl,  
Fax (+ 31-70) 340-3016

Authorized officer

Verhoof, P

# INTERNATIONAL SEARCH REPORT

Inter- national Application No  
PCT/US 93/12416

## C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
P,X	<p>38TH ANNUAL IEEE COMPUTER SOCIETY INTERNATIONAL COMPUTER CONFERENCE - COMPCON SPRING '93, 22-26 FEB 1993 SAN FRANCISCO, CA, USA, 1993, IEEE, PISCATAWAY, USA. pages 441 - 447 P. KNEBEL ET AL. 'HP's PA7100LC: a low-cost superscalar PA-RISC processor' * section 3.2 *</p> <p style="text-align: center;">-----</p>	1-14

Form PCT/ISA/210 (continuation of second sheet) (July 1992)

# INTERNATIONAL SEARCH REPORT

information on patent family members

International Application No

PCT/US 93/12416

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US-A-5107415	21-04-92	JP-A- 2113381	25-04-90
EP-A-0282969	21-09-88	JP-A- 64001050	05-01-89
EP-A-0470570	12-02-92	JP-A- 6124201	06-05-94

Form PCT/ISA/210 (patent family annex) (July 1992)

**THIS PAGE BLANK (USPTO)**